Departamento de Informática

PROYECTO FIN DE CARRERA

Evolución de árboles en el problema del wall following robot

Autor: David Fernández García
Tutor: César Estébanez Tascón

Leganés, 13 de marzo de 2012

Título: EVOLUCIÓN DE ÁRBOLES EN EL PROBLEMA DEL WALL FOLLOWING ROBOT

Autor: David Fernández García

Director: César Estébanez Tascón

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 13 de marzo de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Una persona es uno mismo y sus circunstancias, por ello quiero agradecer en este trabajo a todas aquellas personas que han estado a mí alrededor durante todo el tiempo que me ha llevado concluir este proyecto.

En primer lugar quiero agradecer a Diana su tiempo, dedicación, paciencia e interés, aún cuando le podría estar sonando a “chino” todo aquello que le contaba, y especialmente su capacidad de no tirar la toalla y dejarme por imposible.

También quiero agradecer a mi familia su apoyo, paciencia e insistencia durante todos estos años, hasta que finalmente terminé el proyecto.

Por supuesto, entre los agradecimientos están incluidos todos mis amigos, lo sepan o no. Aquellos que siempre han preguntado: ¿cómo va el proyecto? y ¿cuándo lo lees?, aquellos que, sabiéndolo o sin saberlo, siempre han estado ahí.

En los agradecimientos no puede faltar una mención a todos los profesores que he tenido a lo largo de mi vida, no en balde buena parte de lo que soy, de aquello a lo que he llegado, se lo debo a ellos.

Resumen

Este proyecto trata la hipótesis de que existe una relación entre forma arbórea de los individuos y su adaptación a las soluciones, permitiendo estimar el fitness de los individuos en función de su forma arbórea.

Para ello se ha implementado el problema del wall following robot, según lo describió John R. Koza en su publicación "Evolution of Subsumption Using Genetic Programming". Finalmente se comentarán las conclusiones extraídas a partir de los resultados obtenidos del motor de programación genética ProGen. Estos resultados constarán de una comparación de rendimiento entre la programación genética en árbol y la programación genética tradicional, evaluando tanto los resultados brutos, como el coste empleado para extraerlos.

Por último se implementó una herramienta de visualización de resultados, que permite poder interpretar de forma intuitiva los experimentos. El movimiento de los robots se representa como una línea azul dentro de la habitación de paredes rojas como se muestra en las distintas capturas de pantalla de este documento.

Palabras clave: Este proyecto trata la hipótesis de que existe una relación entre forma arbórea de los individuos y su fitness.

Abstract

This Project deals with the hypothesis that there is a relationship between the individual's tree form and their way to find a solution, allowing us to estimate the fitness value of the individuals based on their tree form.

In order to fulfill the hypothesis this Project has developed the Wall Fallowing Robot problem as described by John R. Koza in his paper "Evolution of Subsumption Using Genetic Programming". The ProGen genetic programming engine was used to achieve this and obtain the results discussed in the conclusions chapter of this memo.

Finally, a viewer was developed to be able to easily show the results of each experiment. In this viewer the path followed by the robots is represented by a blue line inside of a red square room. Screen captures of this viewer have been included in this document.

Keywords: This Project deals with the hypothesis that there is a relationship between the individual's tree form and its fitness value.

Índice de contenidos

Índice de imágenes

Índice de tablas

1. Presentación

En este proyecto se va a presentar una comparativa entre uno de los paradigmas más utilizados de la inteligencia artificial, la programación genética, con una nueva faceta orientada a optimizar dicho paradigma: la programación genética en árbol.

La comparativa se llevará a cabo utilizando el motor de programación genética ProGen desarrollado por alumnos de esta misma universidad bajo la dirección del profesor César Estébanez Tascón. Dicho motor de programación genética se caracteriza por disponer de las herramientas necesarias para el desarrollo y testeo de la programación genética en árbol.

Para realizar la comparativa se utilizará un problema conocido dentro de la inteligencia artificial: el problema “Wall Following Robot”.

Durante el estudio se presentarán varias ejecuciones de este problema en igualdad de condiciones tanto para la programación genética tradicional como para la programación genética en árbol y se extraerán las conclusiones que reflejen los experimentos realizados.

A continuación se presentará una breve introducción a los fundamentos tanto de la programación genética como de la arbórea, así como al motor de programación genética que se ha utilizado en la ejecución de las pruebas.

Posteriormente se mostrará un análisis de los resultados obtenidos durante las distintas ejecuciones y finalmente se ofrecerá el análisis comparativo de ambos paradigmas de inteligencia artificial.

2. Introducción

2.1. Introducción a la Programación Genética

A continuación se presentará una breve introducción a la programación genética. Una vez se hayan aclarado los aspectos más importantes de la misma, se procederá a explicar los fundamentos y las razones que han llevado al desarrollo de la programación genética en árbol. Por último se describirán las características del motor de programación genética que se ha utilizado durante la ejecución de las pruebas.

2.1.1. Programación Genética

La programación genética es uno de los paradigmas de inteligencia artificial que más impulso está recibiendo en los últimos años. Esto se debe entre otras razones a los buenos resultados que ofrece en la resolución de problemas no triviales.

La programación genética se basa en el modelo definido por la selección natural que ha utilizado la naturaleza durante toda la evolución de las especies. En este paradigma cada solución a un problema se describe como una secuencia de pasos sencillos que combinados ofrecen una solución. Esta codificación representa el papel del ADN en los seres vivos.

El ADN de las especies ha ido cambiando a lo largo de la historia, modificando su estructura para adaptarse de la mejor forma posible al entorno en el que se encuentran. Aquellas especies que no han conseguido adaptarse al medio han terminado desapareciendo, dejando paso a las especies que han evolucionando adaptándose de mejor manera a las necesidades que define el entorno.

Este principio de la selección natural es en el que se basa la programación genética, generando conjuntos de soluciones que van evolucionando adaptándose al problema que se desea resolver. Las soluciones que más se acerquen a la solución ideal son replicadas, mientras que las peores soluciones son desechadas produciéndose siempre un avance hacia la mejor solución posible.

2.1.2. Los individuos

Las soluciones, individuos a partir de ahora, son estructuras de información arbóreas que resuelven de un modo, más o menos acertado, un problema. Las estructuras arbóreas son estructuras jerárquicas de nodos de la siguiente forma:

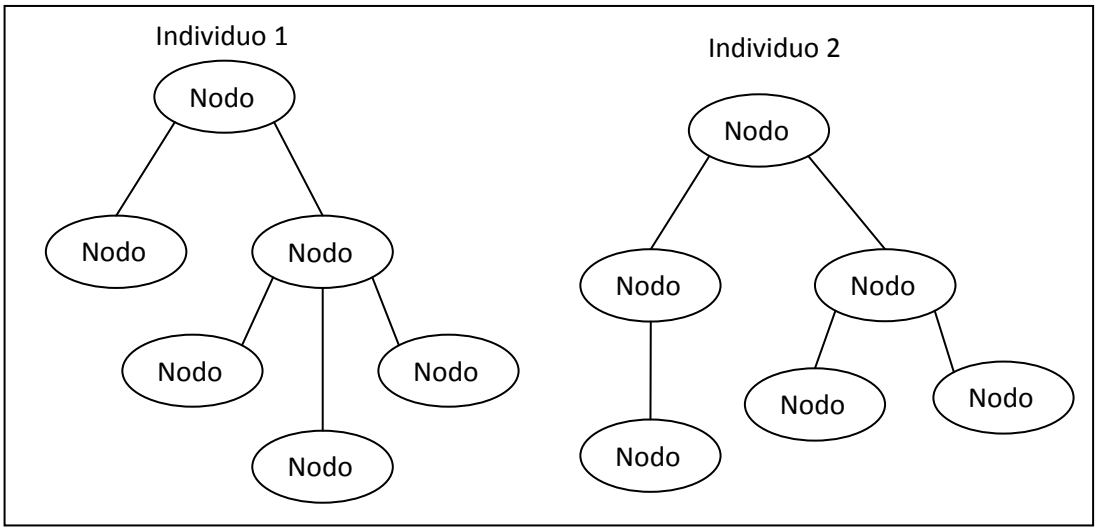


Ilustración 1 - Ejemplo de individuos

Los individuos están formados por un conjunto de nodos relacionados entre sí. Los nodos pueden ser terminales u operadores. Un nodo terminal es un elemento que contiene información en sí mismo y es independiente del resto de nodos, mientras que un operador es un nodo cuyo valor se debe resolver a partir de una serie de nodos de entrada. El número de entradas de un operador se conoce como aridad del nodo. A continuación se ofrece un ejemplo de distintos nodos

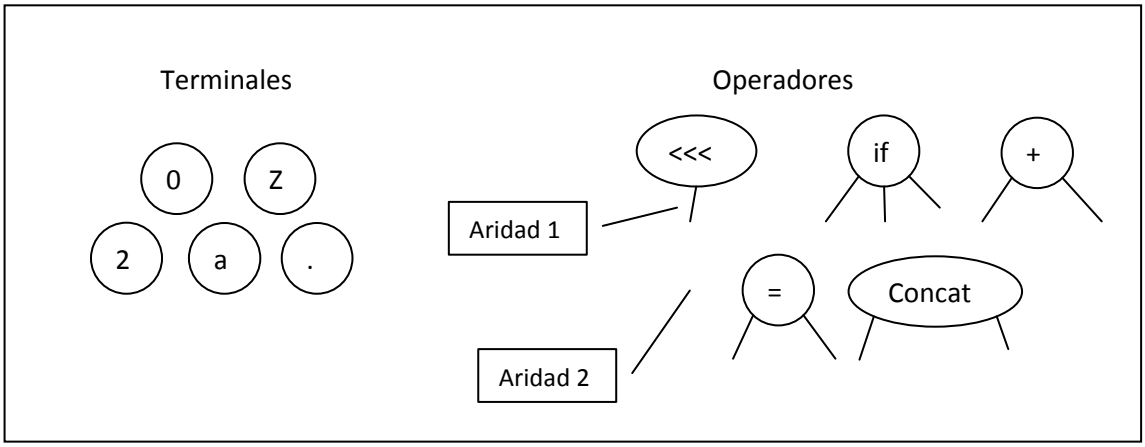


Ilustración 2 - Ejemplo de nodos

Los individuos, son una concatenación de operadores y terminales que ofrecen una solución a un problema. Un ejemplo de individuo con los operadores mencionados anteriormente sería:

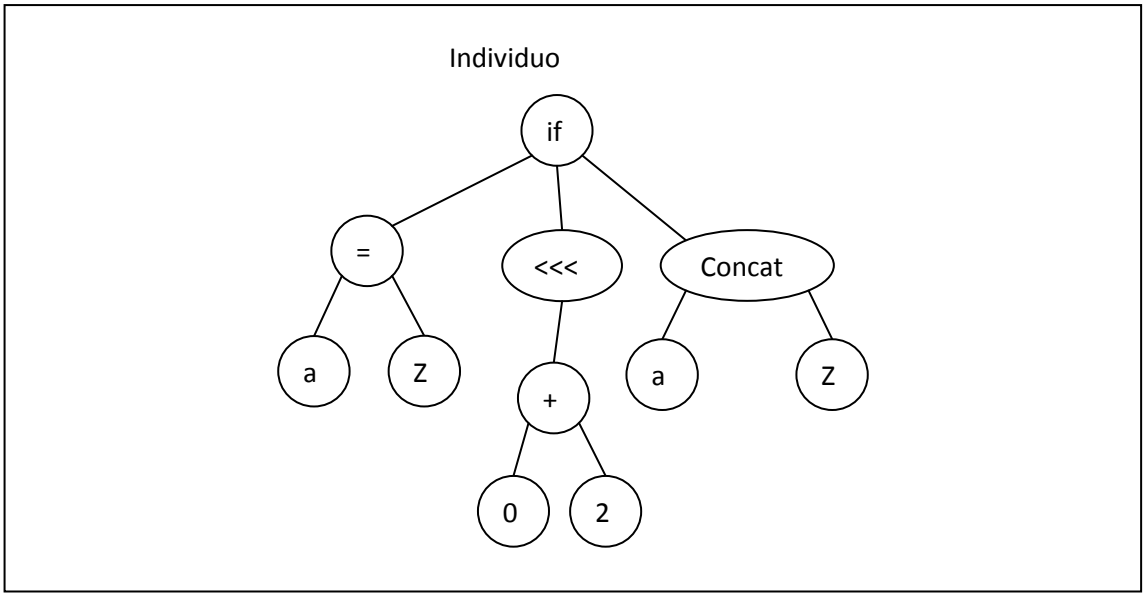


Ilustración 3 - Ejemplo de individuo concreto

La interpretación de los individuos se lleva a cabo mediante una lectura pre-orden de los nodos que conforman el individuo, interpretando cada una de las ramas que cuelgan de un operador en función de la naturaleza del propio operador. Es decir, un operador de suma devolverá el resultado de la operación suma sobre cada una de las entradas que reciba, mientras que, por ejemplo, un operador "if" evaluará de forma lógica la primera de las ramas y si el resultado es verdadero ejecutará la segunda de sus ramas y en caso contrario la tercera.

De este modo, la interpretación de alto nivel del individuo de ejemplo anteriormente mostrado sería:

```
si ( "a" es igual a "Z" )
entonces devuelvo
   el desplazamiento lógico a la izquierda de
      "0" sumado con "2"
si no devuelvo
   "a" concatenado con "Z"
```

Ilustración 4 - Interpretación del individuo

Una vez que se ha ilustrado la representación de los individuos de la programación genética, procederemos a explicar el mecanismo de la misma. En este punto quiero reseñar que se ha ofrecido una vista general de la representación de soluciones en la programación genética, puesto que este punto ofrece problemas complicados de resolver.

Entre los problemas más habituales está la tipología de operaciones y terminales, puesto que pueden ser numéricos, lógicos o alfabéticos, no siempre es fácil determinar el modo de resolver un nodo operador si, como hijos, puede recibir valores de distintos tipos (ej., números, letras, etc.). De este punto han surgido dos ramas de programación genética: la programación genética tipada y la programación genética no tipada.

Otro aspecto crítico es la elección de terminales y operadores que permitan alcanzar la solución óptima del problema, teniendo en cuenta que cuanto mayor sea el número de operadores y terminales mayor será la complejidad del problema.

Menciono estos dos casos por mencionar sólo los primeros problemas que aparecen siempre que se quiere modelar un problema en programación genética, aunque según el problema que se desee modelar pueden aparecer otros muchos.

2.1.3. Las poblaciones

Para resolver un problema se genera, aleatoriamente, una población inicial de un número determinado de individuos. Como en la naturaleza estos individuos deben evolucionar adaptando su ADN (estructura arbórea interna de operadores y terminales) a la mejor forma de resolver el problema modelado.

La ejecución de la programación genética sigue, principalmente los siguientes pasos:

1. Generación de la población inicial
2. Evaluación de la población actual
3. Evolución de la población actual
4. Detectar fin de la ejecución o volver a 2

La población inicial se obtiene generando un número de individuos al comienzo de la ejecución. En la generación de los individuos iniciales se ha de prestar especial atención a que dichos individuos tengan una estructura válida.

En la fase de evaluación de la población se recorre la misma, evaluando cada uno de los individuos, asignándoles una puntuación (fitness) a cada uno de ellos. Esta puntuación es el factor más importante de la programación genética, puesto que debe ser capaz de orientar la evolución de las poblaciones hacia la solución del problema. Existen distintas medidas de fitness "estándar", por decirlo de algún modo. Las más habituales son el Raw Fitness, que es la puntuación que el problema asigne al individuo, y el Adjusted Fitness, cuyo valor debe ser 0 para la solución del problema y 1 para la peor solución posible. De esta forma, cuanto más próximo a 0 sea el Adjusted Fitness, mejor solución representa el individuo.

La fase de evolución de la población consta de 2 subfases. En la primera subfase se realiza la selección del individuo o individuos sobre los que se aplicará el operador genético que se vaya a aplicar, y en la segunda subfase se aplica el operador genético.

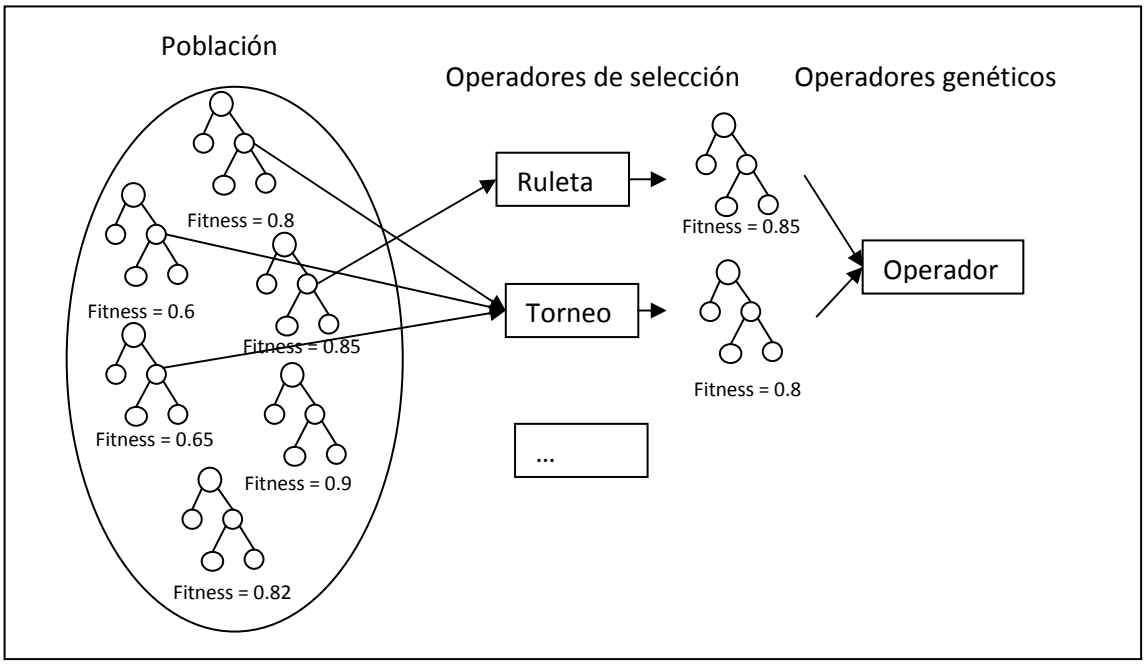


Ilustración 5 - Diagrama de la evolución

La selección de los individuos se realiza mediante operadores de selección que ofrecen distintas técnicas de selección. Ejemplos de operadores de selección serían la ruleta o torneo. La ruleta asigna a cada individuo un número de probabilidades de ser elegido proporcional a lo bueno que sea su fitness, de esta forma los mejores individuos tienen más probabilidades de ser elegidos y pasar a la siguiente generación. El operador de torneo, selecciona una serie de individuos aleatoriamente y finalmente selecciona el que tenga mejor fitness de ellos.

Como se ha observado, los operadores de selección tienden a seleccionar los mejores individuos de la población para replicarse entre ellos, produciendo, presumiblemente, al menos individuos tan buenos o mejores que ellos mismos. De esta forma cada vez se obtendrán más individuos mejor adaptados al problema (mejores soluciones), mejorando la población.

2.1.4. Los operadores de selección

Mediante los operadores de selección se maneja la presión selectiva, es decir, si un individuo debe ser mucho mejor que otros para que sea seleccionado. Un ejemplo muy claro de esto es el operador de Torneo. Si el torneo tiene tamaño 1, aunque el individuo sea el peor de la población pasará a la siguiente generación, mientras que si el tamaño es 10, un individuo malo tiene muy pocas posibilidades de pasar a la siguiente generación, porque es muy fácil que alguno de los otros 9 individuos del torneo sea mejor que él.

Los operadores de selección representan las probabilidades que tiene un individuo de llegar a reproducirse. Se ha de tener en cuenta que, siguiendo las directrices de la selección natural, cuanto mejor sea un individuo más probabilidades tiene de reproducirse.

Una vez se han seleccionado los individuos que se necesitan para el operador genético, este generará una serie de individuos, en función de las características del operador, que pasarán a formar parte de la siguiente generación.

Los operadores de selección más habituales son la ruleta y el torneo, aunque existen más que son poco utilizados o se han dejado de utilizar.

- Elitismo: el operador de elitismo consiste en dar a los mejores individuos de la generación un acceso directo a la siguiente generación. Es decir, estos individuos no se reproducen, sino que pasan directamente a la siguiente generación. Este operador debe utilizarse con precaución porque si son muchos los individuos que se mantienen entre generaciones puede detener la evolución.

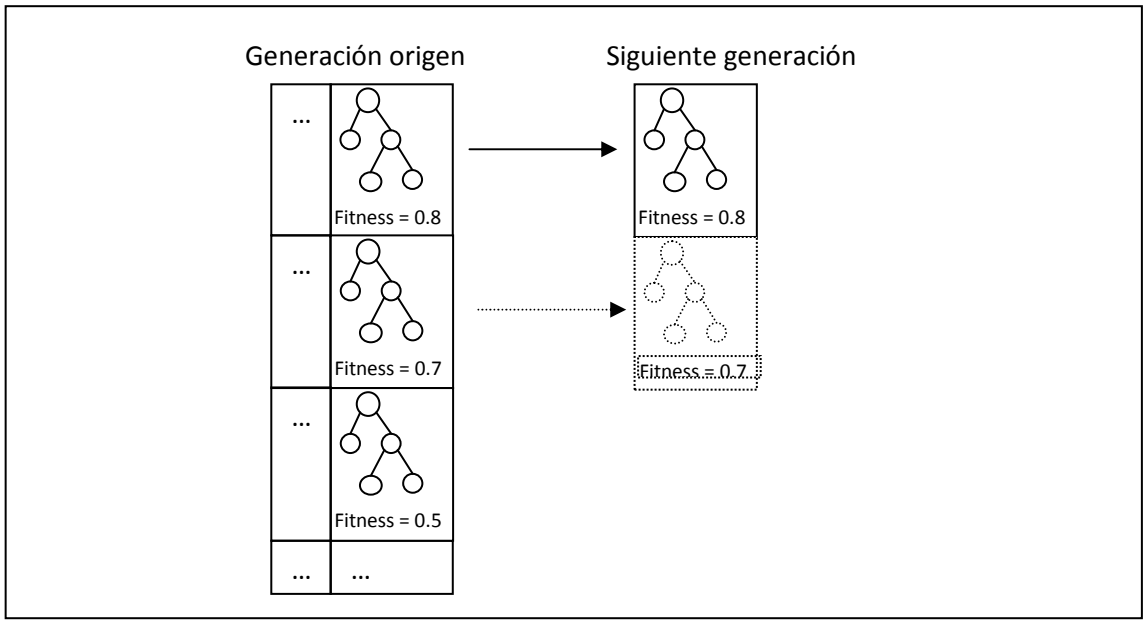


Ilustración 6 - Operador de elitismo

- Ruleta: el operador de ruleta consiste en dar, a cada individuo de la población, una probabilidad de pasar a la siguiente proporcional a su fitness. De esta forma, un mejor individuo tiene más probabilidad de pasar a la siguiente generación que uno peor.

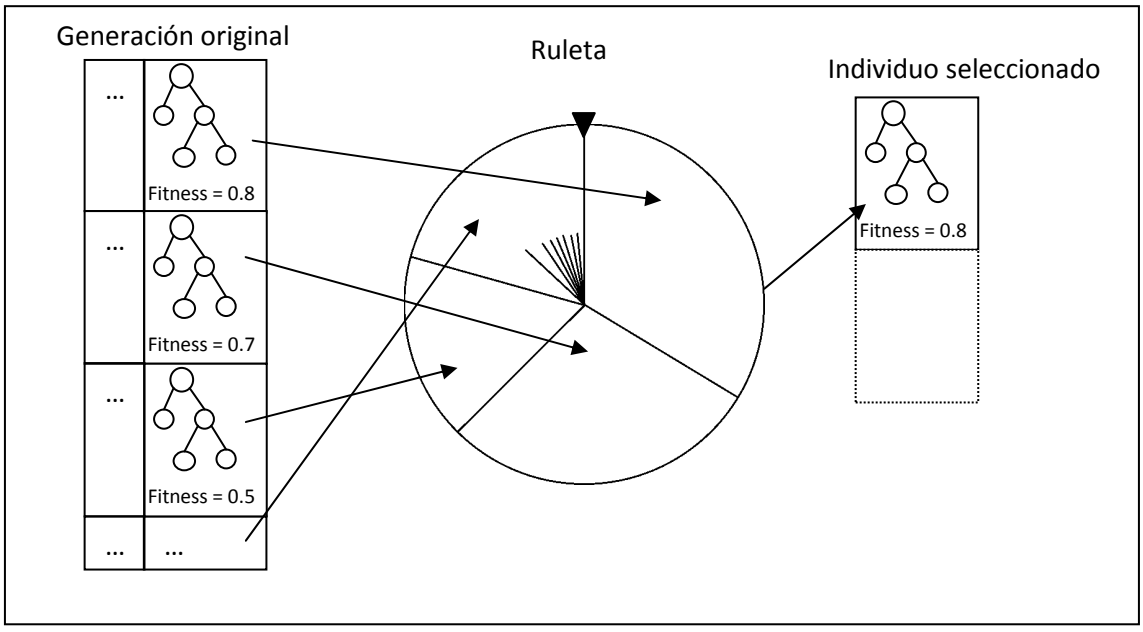


Ilustración 7 - Operador de ruleta

- Ranking: El operador de ranking es una modificación de la ruleta que se utiliza en casos en los que los individuos tienen valores fitness muy similares. Asigna una probabilidad de ser elegido fija en función de su posición en la población, en lugar de en función de su fitness.

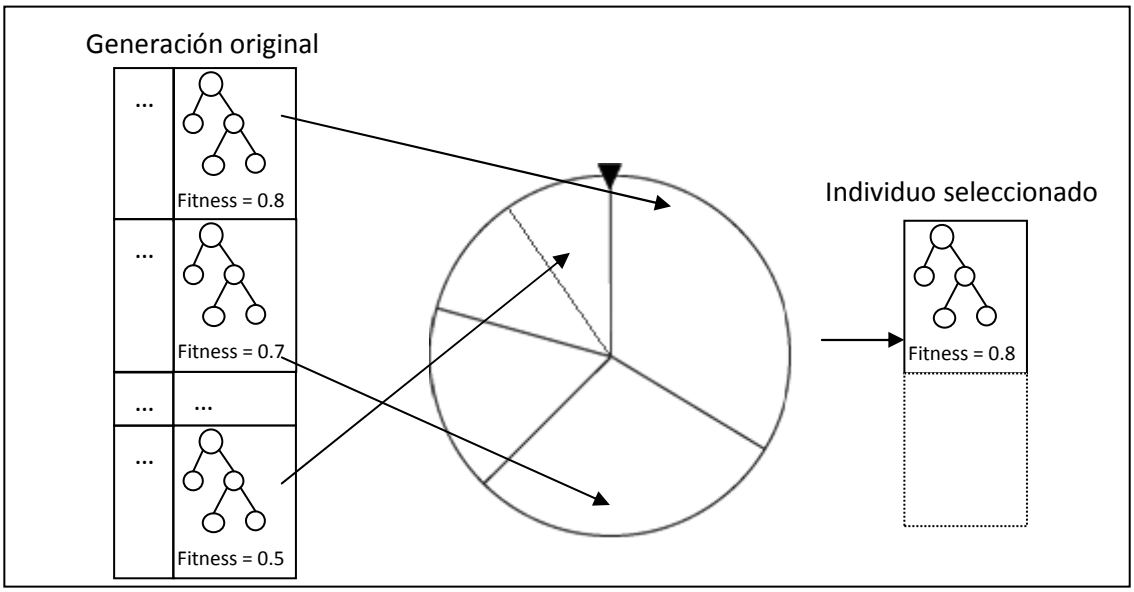


Ilustración 8 - Operador de ranking

- Torneo: el operador de torneo consiste en seleccionar de forma aleatoria un número de candidatos y pasar a la siguiente generación el mejor de entre los candidatos.

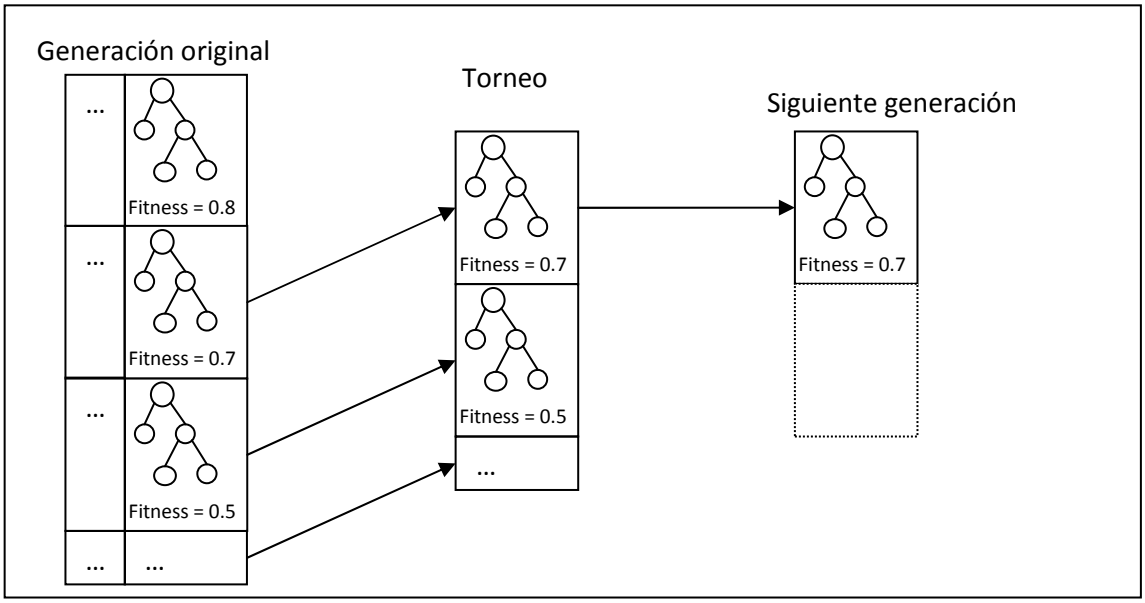


Ilustración 9 - Operador de torneo

2.1.5. Los operadores genéticos

Los operadores genéticos son los encargados de generar los individuos para la nueva población. Estos operadores representación la función que, en la naturaleza, es la reproducción, preservando el ADN (estructura de datos) de los mejores individuos de una generación a otra.

Los operadores más habituales son cruce y mutación, teniendo cada uno de ellos una gran cantidad de variantes.

- Cruce: el operador de cruce es una implementación de la reproducción. Requiere 2 individuos que harán las veces de padres y generarán otros tantos individuos para la nueva generación. Los descendientes tienen parte de la estructura de cada uno de los padres.
    - Cruce simple: se hace un corte en la estructura de los padres y se combinan.

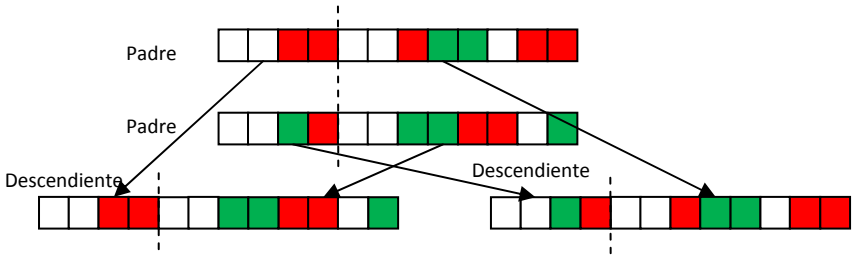


Ilustración 10 - Operador de cruce simple

    - Cruce multipunto: es igual que el cruce simple, solo que se realiza más de un corte en la estructura el individuo.

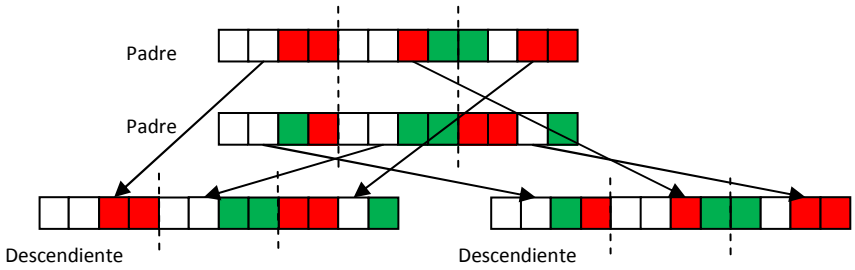


Ilustración 11 - Operador de cruce multipunto

    - Cruce uniforme: siempre pasan los mismos elementos del primer padre al primer descendiente, y los complementarios al segundo descendiente, y a la inversa con el segundo padre.

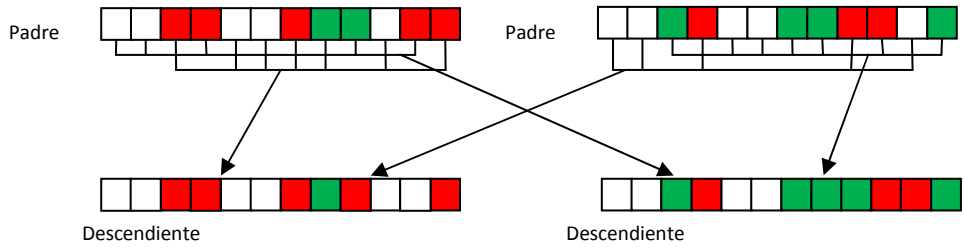


Ilustración 12 - Operador de cruce fijo

- Mutación: la mutación debe realizarse con cuidado, puesto que demasiada mutación convertiría la evolución en un proceso aleatorio. La mutación transforma un elemento en otro distinto. La principal función de la mutación es evitar soluciones que representen un máximo local y evitar que todos los individuos terminen siendo exactamente iguales. También hay varios tipos de mutación, pero la más utilizada es la mutación sobre un punto.

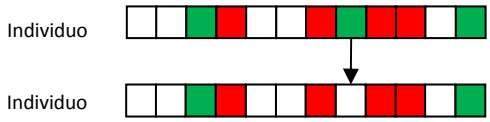


Ilustración 13 - Operador de cruce fijo

- Inversión: cambia el orden de los elementos de un individuo.

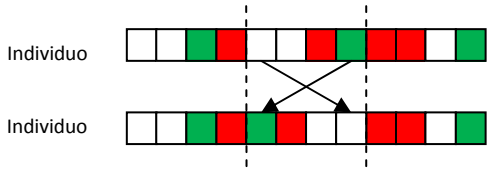


Ilustración 14 - Operador de inversión

Finalmente, con la ejecución de los operadores genéticos se ha obtenido una nueva población (nueva generación) sobre la que se repetirá el proceso de evolución, hasta que uno de los individuos sea una solución perfecta del problema, o se hayan completado el total de generaciones definidas.

2.2. Programación genética en árbol

2.2.1. Planteamiento

La programación genética tradicional plantea los individuos como una secuencia de pasos que llevan una situación inicial a un punto más próximo a la solución deseada de lo que se encontraba inicialmente.

En este planteamiento el peso de la evolución, de la resolución del problema, recae en el orden en que se ejecuten los operadores, siendo esto definido por la forma del árbol y por la posición de los propios operadores dentro del árbol.

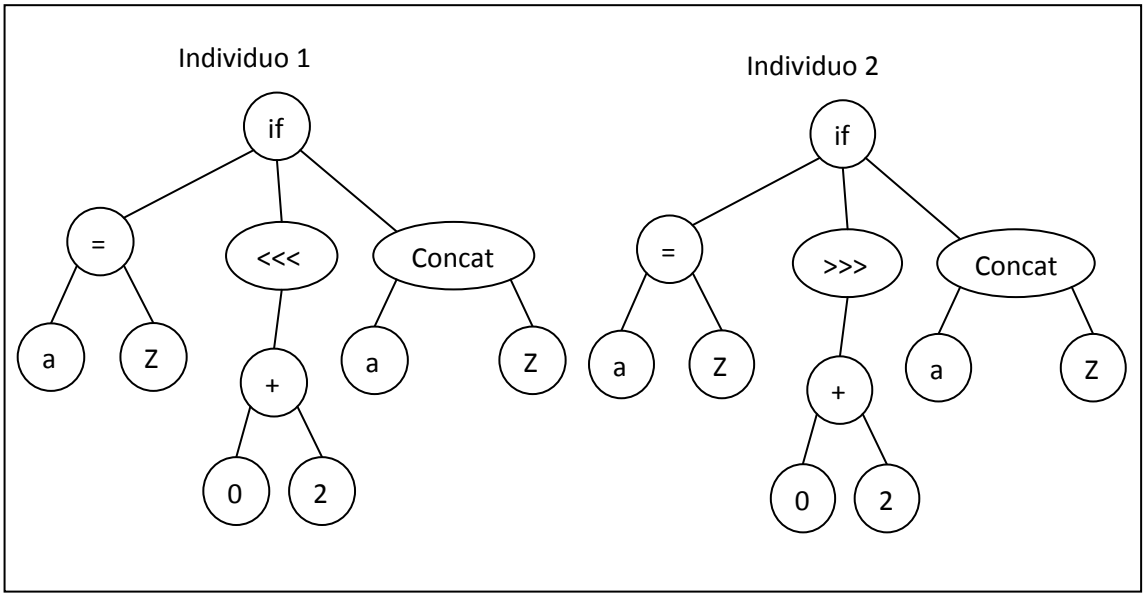


Ilustración 15 - Soluciones representadas por individuos

En este caso, se puede observar como el individuo 1 y el individuo 2 plantean dos caminos diferentes hacia la solución.

En concreto, el individuo 1 plantea un desplazamiento lógico hacia la izquierda en caso de que a sea igual a Z, mientras que el individuo 2 plantea un desplazamiento lógico hacia la derecha en ese mismo caso.

Para este ejemplo, tan solo ha cambiado un operador, pero podrían haber cambiado tantos como se hubiese deseado, proporcionando infinidad de soluciones diferentes. Esto no sólo es aplicable a los nodos, sino que la propia estructura del árbol también determina el orden en el que se ejecutaran los nodos.

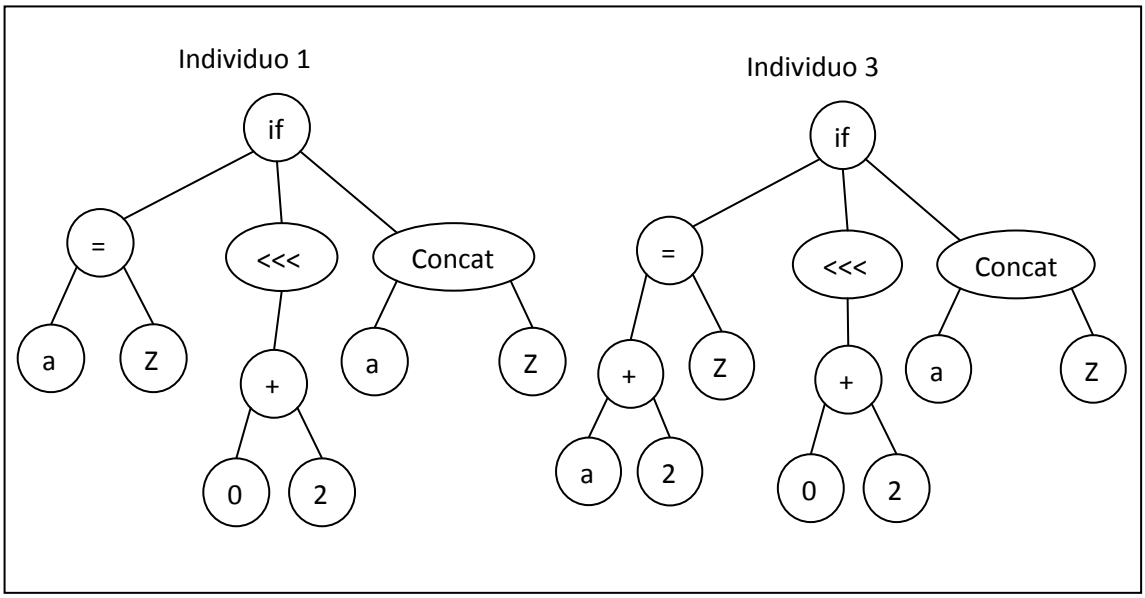


Ilustración 16 - Cambio de solución al cambiar la forma del árbol

En este caso se puede observar que cambiando la forma del árbol, también ha cambiado el camino hacia la solución que definen cada uno de los individuos. El primer individuo sigue siendo el que hace un desplazamiento lógico hacia la izquierda en caso de que a sea igual a Z, mientras que el individuo 3 hace el desplazamiento lógico a la izquierda en caso de que se cumpla la condición “a + 2 = Z”.

En este caso, cambiar la forma del árbol, también ha requerido cambiar las nodos, puesto que la aridad de los mismos está directamente relacionada con la estructura del árbol, sin embargo queremos resaltar el cambio en la estructura del árbol, porque en esto se fundamenta la programación genética en árbol. Esta programación nos permitirá modificar la estructura del árbol ignorando los operadores genéticos, dentro de una misma aridad.

La programación genética en árbol permite evolucionar la forma de los árboles independiente de los nodos que los compongan. La primera evoluciona las formas de los árboles en forma de individuos arbóreos. Con estos individuos arbóreos se compone una población inicial óptima para la segunda parte de la evolución, que consiste en programación genética tradicional. En esta parte se evoluciona, principalmente, los operadores genéticos con poblaciones iniciales generadas a partir de los árboles óptimos obtenidos de la sección de la evolución arbórea, aunque puede continuar modificando la estructura de los árboles.

Cabe destacar que la idea de la programación genética en árbol no es aportar soluciones, sino poblaciones iniciales optimizadas de forma que la programación genética obtenga buenas soluciones con un coste mínimo.

La programación genética en árbol se fundamenta en la defensa de la hipótesis de que la estructura del árbol de un individuo influye directamente en su grado de adaptación a la solución deseada, independientemente de que entre todos los árboles definidos por una estructura haya unos individuos con mejor adecuación al problema que otros. Es decir, plantea que a partir de dos formas distintas de árbol de dos individuos, se puede estimar estadísticamente cuál de ellos es mejor, sin necesidad de evaluar todas las combinaciones posibles de individuos.

Intuitivamente se puede justificar esta hipótesis con un ejemplo muy sencillo. Supongamos que nuestro problema es recorrer un pasillo. Los operadores disponibles son concatenar acciones, si-menor, avanzar y girar. Para ilustrar este caso vamos a seleccionar los casos extremos de árboles. Uno en profundidad y otro en anchura.

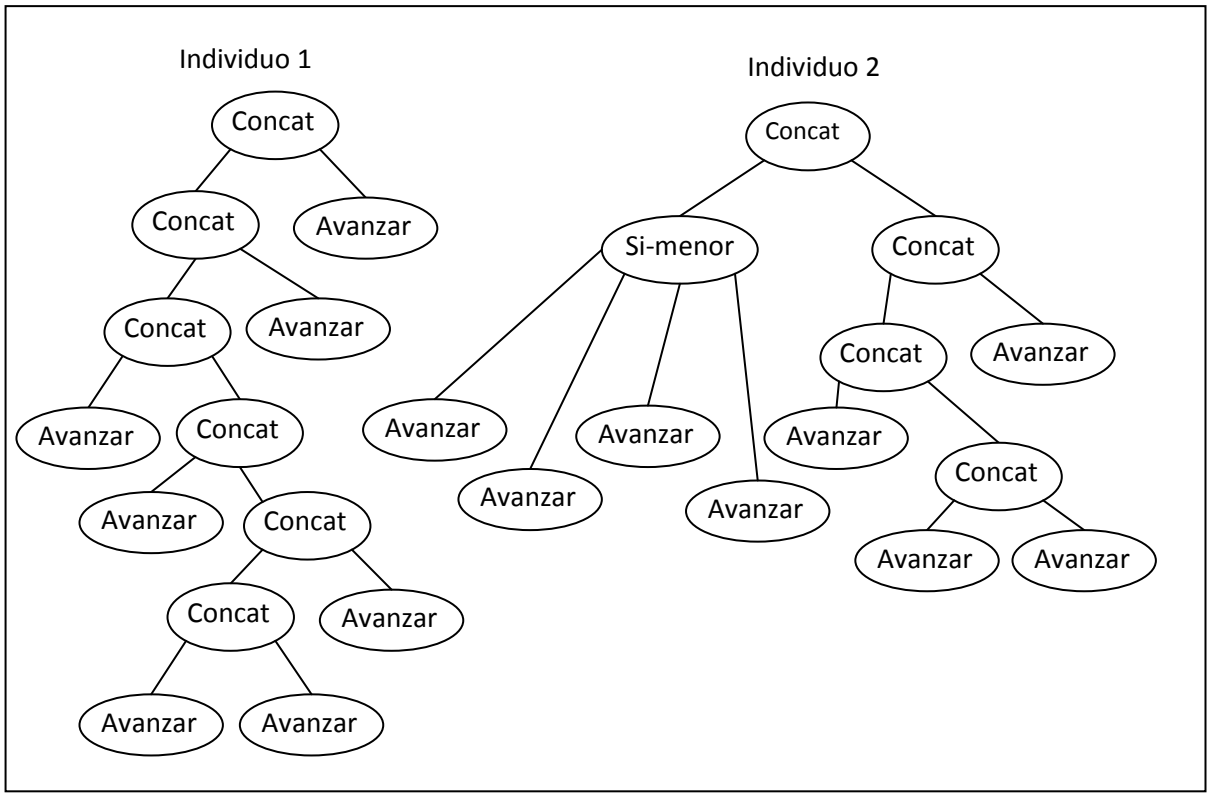


Ilustración 17 - Influencia de la forma del árbol en los individuos

En lo referente a nodos, los dos individuos mostrados arriba tienen exactamente el mismo número de nodos: 13. Cambiar la forma del árbol requiere cambiar la aridad de algunos nodos, y con ello cambiar dichos operadores, pero para este caso igual que se ha usado si-menor, podría haber sido si-mayor, si-igual y los resultados serían los mismos.

El individuo 1 tiene en 6 de sus nodos el operador concat y en 7 el operador Avanzar, mientras que Individuo 2 tiene 4 operadores concat, 1 operador si-menor y 8 operadores avanzar, sin embargo, la estructura del individuo 1 hace que sea mejor que el otro. En este punto hay que recordar que hablar de la estructura del árbol es hablar de los nodos intermedios e ignorar los terminales y por ello todo lo que se diga es dependiente de cada problema, puesto que depende de los operadores.

En esta representación, se han puesto todos los terminales avanzar, aunque podrían haber sido también girar, pero con este ejemplo se maximiza la diferencia. Con estos individuos de la ilustración, se obtendrían los siguientes resultados:

Individuo 1: avanzar 7 casillas del pasillo.

Individuo 2: avanzar 5 casillas del pasillo.

Como se ha comentado anteriormente, este es el caso que maximiza la diferencia, pero en esta parte entran en juego las estadísticas.

Individuo 1:

- Con 0 giros
  - 100% de probabilidades de avanzar 7
- Con 1 giro
  - 14,28% de probabilidades de avanzar 6
  - 14,28% de probabilidades de avanzar 5
  - 14,28% de probabilidades de avanzar 4
  - 14,28% de probabilidades de avanzar 3
  - 14,28% de probabilidades de avanzar 2
  - 14,28% de probabilidades de avanzar 1
  - 14,28% de probabilidades de avanzar 0
- Con 2 giros…

Individuo 2:

- Con 0 giros
  - 100% de probabilidades de avanzar 5.
- Con 1 giro
  - 37,5% de probabilidades de avanzar 5.
  - 12,5% de probabilidades de avanzar 4
  - 12,5% de probabilidades de avanzar 3
  - 12,5% de probabilidades de avanzar 2
  - 12,5% de probabilidades de avanzar 1

- 12,5% de probabilidades de avanzar 0.

- Con 2 giros…

Esta estimación es fácilmente justificable, veamos el caso del individuo 1 con un giro.

La lectura (in orden) del individuo, que es como se suele trabajar en estos casos, y es la forma en que trabaja ProGen y que explicaré en el apartado 1.3, es la siguiente: Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.

Cuando se introduce un giro, este tiene la misma probabilidad de caer en un avanzar que en otro, quedando de la siguiente forma:

Programa del individuo	Casillas que avanza
Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Girar.	6
Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Girar, Avanzar.	5
Avanzar, Avanzar, Avanzar, Avanzar, Girar, Avanzar, Avanzar.	4
Avanzar, Avanzar, Avanzar, Girar, Avanzar, Avanzar, Avanzar.	3
Avanzar, Avanzar, Girar, Avanzar, Avanzar, Avanzar, Avanzar.	2
Avanzar, Girar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.	1
Girar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.	0

El caso del individuo 2 es algo más complejo, por su operador si-menor.

Hijos de si-menor	Resultado
Avanzar, Avanzar, Avanzar, Girar.	Girar
Avanzar, Avanzar, Girar, Avanzar.	Avanzar
Avanzar, Girar, Avanzar, Avanzar.	Avanzar
Girar, Avanzar, Avanzar, Avanzar.	Avanzar

Esto hace que la tabla del individuo resulte de la siguiente forma:

Programa del individuo	Avance
(Avanzar, Avanzar, Avanzar, Avanzar) -> Avanzar, Avanzar, Avanzar, Avanzar, Girar.	4
(Avanzar, Avanzar, Avanzar, Avanzar) -> Avanzar, Avanzar, Avanzar, Girar, Avanzar.	3
(Avanzar, Avanzar, Avanzar, Avanzar) -> Avanzar, Avanzar, Girar, Avanzar, Avanzar.	2
(Avanzar, Avanzar, Avanzar, Avanzar) -> Avanzar, Girar, Avanzar, Avanzar, Avanzar.	1

(Avanzar, Avanzar, Avanzar, Girar) -> Girar, Avanzar, Avanzar, Avanzar, Avanzar.	0
(Avanzar, Avanzar, Girar, Avanzar) -> Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.	5
(Avanzar, Girar, Avanzar, Avanzar) -> Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.	5
(Girar, Avanzar, Avanzar, Avanzar) -> Avanzar, Avanzar, Avanzar, Avanzar, Avanzar.	5

Finalmente se puede observar, que pese a que un individuo 1 mostrado sea mejor que el individuo 2 como tal, cuando se introduce un giro, el individuo 2 es estadísticamente mejor que el individuo 1. Por ello se ha de realizar un estudio comparativo para confirmar si se puede justificar la afirmación que plantea la programación genética en árbol.

2.2.2. Individuos arbóreos

Los individuos arbóreos son individuos con la característica de que todos sus nodos son comodines, es decir, que cualquier nodo podrá ser reemplazado posteriormente por otros nodos que tengan la misma aridad.

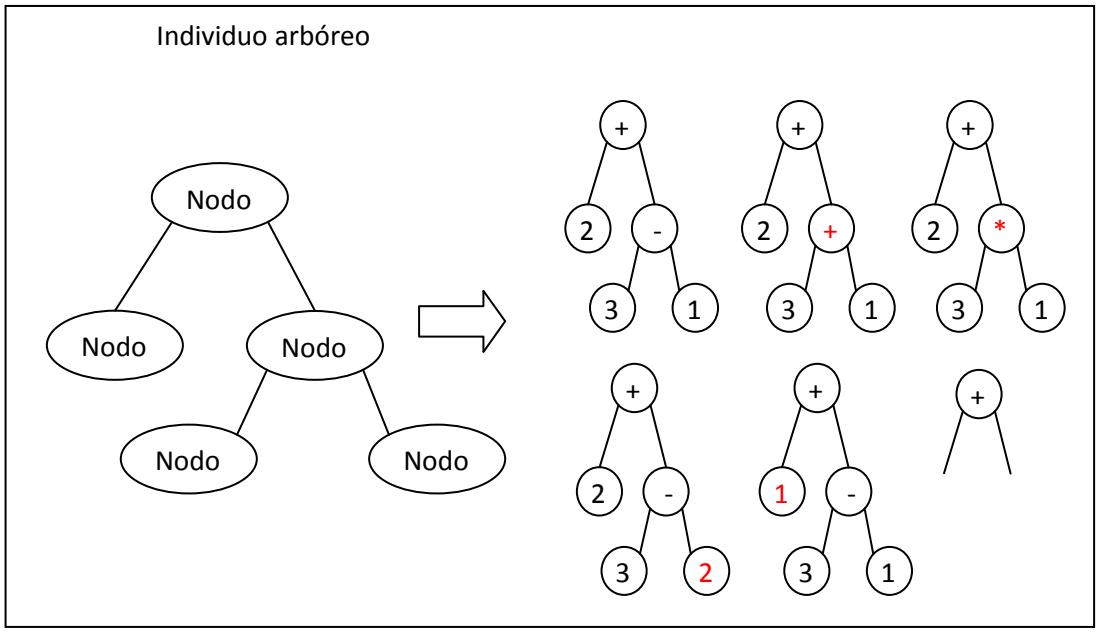


Ilustración 18 - Individuos representados por un individuo arbóreo

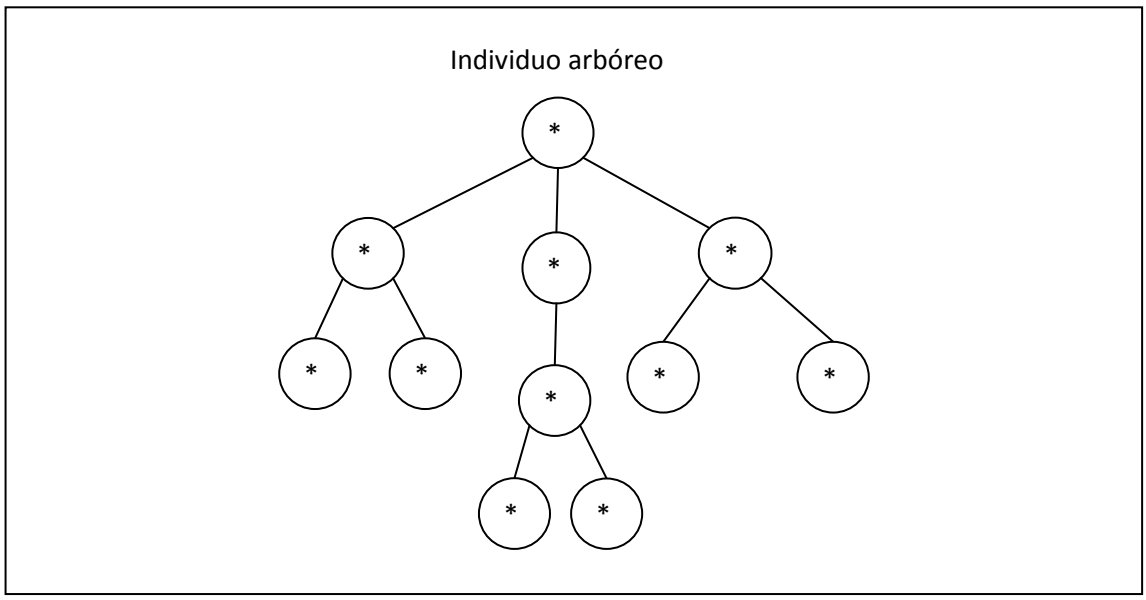


Ilustración 19 - Ejemplo de individuo arbóreo

De esta forma, al evaluar este individuo arbóreo se estará evaluando estadísticamente cómo de buena es la forma del árbol, y todos los individuos

tradicionales con dicha forma, es decir, con el individuo anterior se estima cómo de buenos serán los individuos que coincidan con esa forma de árbol, como pueden ser:

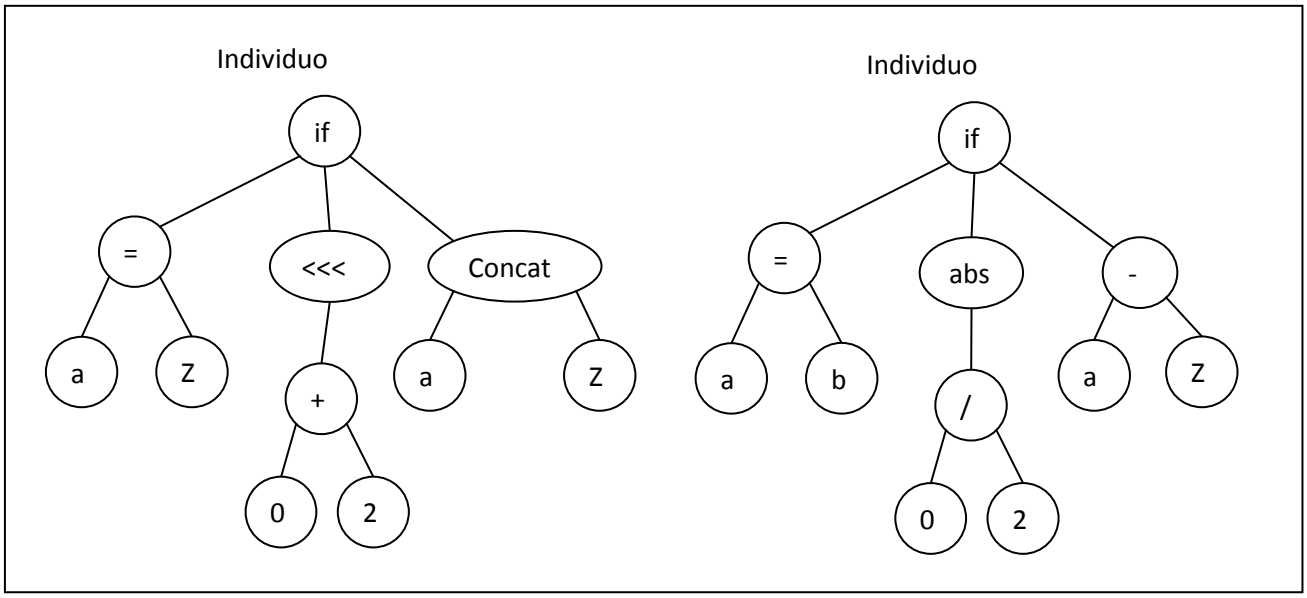


Ilustración 20 - Ejemplo de individuos representados por el individuo arbóreo

La programación genética en árbol consta de las mismas partes de la programación genética tradicional:

- Construcción de la población inicial
- Evaluación de la población
- Selección de los mejores individuos
- Evolución de los mejores individuos

Como ya se ha comentado anteriormente, el objetivo de la programación genética en árbol no es proporcionar soluciones al problema, por lo que la condición de parada no se basa en el fitness de los individuos arbóreos, además de que este no suele ser muy elevado debido a que el fitness de los individuos arbóreos es el fitness medio de los individuos tradicionales que representa.

Por este motivo, se decidió considerar un cierto límite de concurrencia entre los árboles para utilizarlo como condición de parada. Es decir, si fija la condición de parada en el 85%, la fase de programación genética en árbol terminará cuando se acaben las generaciones asignadas o el 85% de los individuos arbóreos que forman la población tengan la misma forma.

Una vez alcanzado este punto, se utilizará el individuo arbóreo con mejor fitness para generar la población inicial de los individuos de programación genética tradicional. Todos los individuos generados tendrán la misma forma de árbol, que es lo que ha

aportado la fase de programación genética en árbol, pero está podrá sufrir pequeños ajustes mediante los operadores genéticos definidos, cruce, mutación, etc.

2.3. ProGen

Un motor de programación genética es un programa encargado de construir una población inicial de individuos con intención de resolver un problema. Una vez se dispone de la población inicial, se hace evolucionar a ésta mediante los operadores genéticos configurados, tratando de emular la evolución de las especies que se ha llevado a cabo en la naturaleza, para obtener unos individuos que sean la solución más óptima de entre todas las disponibles.

ProGen es un motor de programación genética desarrollado en la Universidad Carlos III de Madrid por el profesor D. César Estébanez Tascón. De entre todos los motores de programación genética, éste destaca por ejecutar programación genética tipada.

Como se ha visto en la sección de programación genética, a la hora de construir el árbol que representa un individuo, únicamente se tiene en cuenta la aridad de sus nodos. Esto presenta un problema, porque no todos los nodos reciben por naturaleza el mismo tipo de datos.

La forma más fácil de visualizar esto es mediante un ejemplo. Tomemos el siguiente individuo:

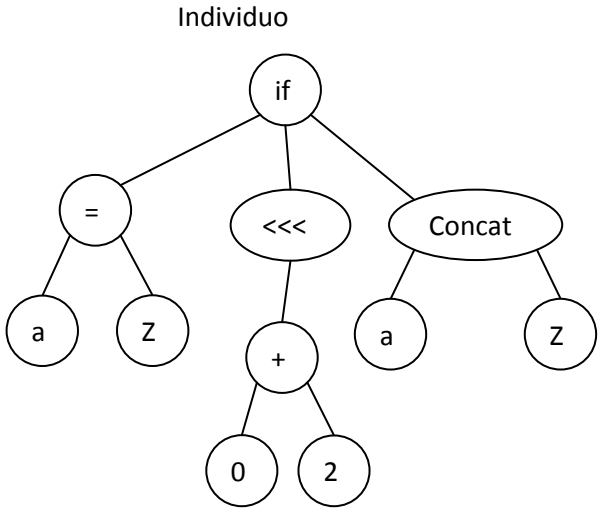


Ilustración 21 - Individuo con problemas de tipado en sus nodos

Como se puede observar en el individuo, su nodo raíz (if) tiene una aridad de 3. El primer hijo es la condición que determinará cuál de las siguientes ramas se ejecutarán. El segundo hijo devuelve una operación lógica realizada sobre el resultado de una suma, lo que es lo mismo, devuelve un número. Y por último, el tercer hijo

devuelve la concatenación de 2 valores, por lo que normalmente se tratará de una cadena de caracteres.

El primer hijo debería devolver un valor booleano para que el if sepa evaluarlo como expresión, pero si esto no se define de ningún modo, podría estar recibiendo un número, una cadena de caracteres, o cualquier resultado que pudiese devolver un operador definido para el problema que se está tratando.

Además de lo descrito anteriormente, si el nodo if tuviese un nodo padre, este recibiría como valor de la evaluación del if un número o una cadena en función del resultado de evaluar el primer hijo del if (la expresión).

Para afrontar este problema se han utilizado dos sistemas:

- Definir un tipo único para todos los tipos de datos con los que puede trabajar el problema, por ejemplo, considerando que el valor booleano falso es el valor numérico 0.

- Programación tipada: Definir los tipos de los parámetros que recibe un operador y su valor de retorno. Así, por ejemplo, el nodo suma quedaría definido de la siguiente forma: “double$$double$$double” y el nodo igual sería “double$$double$$boolean”.

ProGen utiliza este último tipo, añadiendo mayor capacidad a la hora de generar individuos, utilizando para ello gramáticas. Al utilizar programación tipada, podemos despreocuparnos de que un operador matemático reciba como argumento nodos que no sean numéricos.

También añade un nivel más de inteligencia en la generación de individuos y las operaciones realizadas con los mismos, puesto que los nodos que devuelvan valores numéricos, sólo serán utilizados con operadores numéricos, y lo mismo pasará con cada uno de los tipos definidos.

2.3.1. Secuencia de una ejecución

ProGen mantiene la secuencia de ejecución de la programación genética tradicional. Consta de los siguientes pasos:

1. **Generación de la población inicial**

Para la generación de la población inicial, ProGen contaba con un punto de complejidad añadido: la programación tipada.

La generación de individuos es, en realidad, un proceso bastante complejo. Ya no vale insertar cualquier nodo elegido aleatoriamente, puesto que su tipo estará limitado por el valor de entrada o el valor de salida según como se construya el individuo.

Para hacer de ProGen una herramienta realmente útil, también se dotó al sistema de parámetros que permiten definir algunas características que deben tener los individuos con los que nos interesa trabajar. De esta forma, podemos definir una profundidad máxima para los individuos con los que trabajaremos o un número de nodos, según la dirección en la que queramos explorar la solución.

Todas estas características, aunque han aportado un conjunto de posibilidades muy amplio a ProGen, también han supuesto un coste en complejidad bastante elevado.

Para resolver las condiciones impuestas por la programación genética tipada, se han utilizado, en la generación de los individuos, gramáticas, que nos permiten definir cuáles son los nodos válidos para ser hijo del último nodo del árbol.

Por último también se han de resolver las restricciones impuestas por los parámetros definidos para el problema, para lo que se estableció un número de reintentos para la generación de un individuo, de tal forma, que si el individuo generado no cumple las condiciones requeridas, se vuelve a generar otro individuo aleatorio que si que las cumpla. Esto se hace así, aunque sea más costoso, para evitar dirigir la generación de individuos en función de los requisitos, sino dejar que sean completamente aleatorios.

2. **Evaluación de la población actual**

La evaluación de los individuos no requiere ningún procedimiento adicional frente a la programación genética tradicional. Cada problema define su forma de evaluar un individuo, devolviendo un valor numérico que representa el fitness del individuo evaluado.

Como la función de fitness está definida en cada problema, ProGen se puede adaptar a cualquier problema sin que haga falta modificar el motor.

3. Evolución de la población actual

La evolución de las poblaciones se realiza mediante los operadores genéticos definidos en cada problema. Cada problema contará con un archivo de configuración en el que se definirán los operadores genéticos que se utilizan y la probabilidad de aplicar cada uno de ellos. De esta forma, ProGen queda aislado de la implementación específica de cada problema.

Los operadores suelen estar caracterizados por varios elementos, como son: la probabilidad de ser aplicado y el selector a aplicar.

El selector no es otra cosa que un operador que aplicado sobre la población devuelve un subconjunto de tantos individuos como el operador requiera. Los selectores más habituales son ruleta, torneo, etc.

4. Detectar fin de la ejecución

La detección del fin de la ejecución es muy simple, y puede deberse a dos condiciones:

1. El problema ha sido resuelto. Cuando el resultado de la función de fitness de un individuo sea el mejor resultado posible, se obtendrá el resultado de que se ha resuelto el problema planteado, con lo que dicho individuo será la solución al problema.

   Una vez se ha obtenido la solución para el problema, no tiene sentido seguir buscando soluciones, porque ya la tenemos.

   Si se quieren obtener soluciones optimizadas, en la función de fitness se debe haber incluido el criterio de comparación entre soluciones, de forma que el individuo con el mejor fitness posible sea la mejor de entre todas las posibles soluciones.

2. Se ha alcanzado el número máximo de iteraciones. En el problema, se puede definir un número máximo de iteraciones o generaciones en las que se deberá haber obtenido la solución.

   Esto es útil para evitar que el motor de programación genética se quede buscando una solución que no puede obtener al problema.

Las razones que llevan a que una solución sea inalcanzable son variadas, como que los operadores no sean adecuados, que no se dispongan de suficientes recursos o que el tiempo requerido para obtener la solución no sea asumible.

5. Resultados de la ejecución

El sistema de salida de ProGen es un sistema personalizable, es decir, se basa en interfaces para permitir integrar clases realizadas por uno mismo sin necesidad de modificar las clases propias del motor de programación genética.

Además de la posibilidad de añadir clases propias, para el uso estándar de ProGen, se han definido una serie de componentes de salida estándar. Los elementos estándar de salida son principalmente dos: el fichero best_individual.txt y el fichero standar_output.txt.

El fichero best_individual.txt contiene los principales datos del mejor individuo que se ha obtenido durante la ejecución del motor en búsqueda de la solución del problema. La información que muestra tiene el siguiente formato:

```
Best Individual:
        Raw fitness: 49.0
        Adjusted fitness: 0.02

           Tree  Nodes  Depth
        --------------------
         Tree 0     87     13
        --------------------
Tree 0:
  (Prog2   (Prog2   (Prog2   (Prog2   R
          (Prog2   (Prog2   L
              MF
            )
            (Prog2   (ifLwEq   S11
                S5
                MF
                R
              )
              (Prog2   (Prog2   MB
                  (Prog2   MB
                    MB
                  )
                )
                (Prog2   (Prog2   L
                    (Prog2   (ifLwEq   S8
                        S1
                        R
                        (ifLwEq   S9
                          S110
                          (Prog2   MB
                            R
                          )
                          (Prog2   MB
                            R
                          )
                        )
                      )
                      R
                    )
                  )
```

```
                (Prog2   (Prog2   (ifLwEq   S8
                      S4
                      MB
                      MF
                    )
                    (Prog2   MB
                      L
                    )
                  )
                  R
                )
              )
            )
          )
        )
        (Prog2   (Prog2   (ifLwEq   S0
              S2
              MB
              MB
            )
            L
          )
          MB
        )
      )
          (ifLwEq   S11
        S8
        R
        (Prog2   R
          (Prog2   (Prog2   L
              (Prog2   MF
                (ifLwEq   S7
                  S110
                  MB
                  L
                )
              )
            )
            MB
          )
        )
      )
    )
    (Prog2   (Prog2   R
        (Prog2   R
          MF
        )
      )
      (Prog2   L
        (Prog2   MB
          MB
        )
      )
    )
  )
```

Como se puede observar, el archivo contiene información básica sobre el individuo, como es el fitness, tanto absoluto como ajustado y el número de nodos y la profundidad del árbol que describe al individuo.

Finalmente, también se muestra el propio individuo, correctamente indentado para tratar de hacerle lo más legible posible. De forma que el comienzo de la ejecución del individuo mostrado sería:

- Girar a la derecha
- Girar a la izquierda
- Avanzar
- Si sensor 11 menor o igual que sensor 5
    - Avanzar
- Sino
    - Girar a la derecha
- …

El fichero standar_output.txt contiene la traza de la ejecución, es decir, guarda los datos relevantes para cada generación y, a su vez, cada vez que se obtiene un nuevo mejor individuo lo anota, junto con los datos que se han visto en el archivo best_individual.txt. El archivo resultante queda con el siguiente formato:

```
-----------------------
|-=    Generation 0 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             65.0000  |         0.0152  |           63  |            7  |
Generation Mean  |             79.6183  |         0.0124  |      67.0733  |       7.9100  |
  Worst of Gen.  |             80.0000  |         0.0123  |           77  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0000  |                 0.0000  |
Evaluation Time  |              5.4633  |              3278.0000  |
-------------------------------------------------------------------

 New Best Individual:

        Raw fitness: 65.0
        Adjusted fitness: 0.015151515151515152

           Tree  Nodes  Depth
        ---------------------
         Tree 0     63      7
        ---------------------

Tree 0:
        (Prog2  (Prog2  (ifLwEq S0
                                S0
                                (Prog2  (ifLwEq S5
                                                S7
                                                MB
                                                MB
                                        )
                                        R
                                )
                                L
                        )
                        (ifLwEq S7
                                S2
                                MB
                                MB
                        )
                )
                (Prog2  (Prog2  (Prog2  (Prog2  MB
                                                (ifLwEq S3
                                                        S9
                                                        MB
                                                        (Prog2  MB
                                                                MB
                                                        )
                                                )
                                        )
                                )
                                (ifLwEq S9
                                        S9
```

```
                                        (ifLwEq S1
                                                S11
                                                R
                                                MB
                                        )
                                        (ifLwEq S11
                                                S11
                                                MB
                                                (Prog2  L
                                                        R
                                                )
                                        )
                                )
                        )
                        MB
                )
                (Prog2  MB
                        (ifLwEq S7
                                S7
                                MB
                                (ifLwEq S11
                                        S11
                                        L
                                        (Prog2  (ifLwEq S1
                                                        S1
                                                        L
                                                        MB
                                                )
                                                R
                                        )
                                )
                        )
                )
        )
)
```

```
================================================================================================

-----------------------
|-=    Generation 1 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |          35  |           8  |
Generation Mean  |             78.8200  |         0.0125  |     66.8133  |       7.9067  |
  Worst of Gen.  |             80.0000  |         0.0123  |          69  |           7  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0450  |              27.0000  |
Evaluation Time  |              7.0300  |            4218.0000  |
-------------------------------------------------------------------
```

```
 New Best Individual:

        Raw fitness: 55.0
        Adjusted fitness: 0.017857142857142856

           Tree  Nodes  Depth
        ---------------------
         Tree 0     35      8
        ---------------------

Tree 0:
        (Prog2  (ifLwEq S8
                        S3
                        MB
                        (Prog2  (Prog2  MB
                                        MB
                                )
                                (ifLwEq S4
                                        S1
                                        MB
                                        MF
                                )
                        )
                )
                (Prog2  MB
                        (Prog2  (Prog2  (Prog2  (Prog2  MF
                                                        MB
                                                )
                                                (ifLwEq S9
                                                        S8
                                                        MB
                                                        (Prog2  (Prog2  MF
                                                                        MB
                                                                )
                                                                L
                                                        )
                                                )
                                        )
                                        MB
                                )
                                (Prog2  MB
                                        MB
                                )
                        )
                )
        )

================================================================================
```

```
-----------------------
|-=    Generation 2 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |           35  |            8  |
Generation Mean  |             77.0433  |         0.0129  |      69.7433  |       7.8683  |
  Worst of Gen.  |             80.0000  |         0.0123  |           49  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0417  |              25.0000  |
Evaluation Time  |              8.1617  |            4897.0000  |
================================================================================================


-----------------------
|-=    Generation 3 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |           35  |            8  |
Generation Mean  |             73.8650  |         0.0135  |      69.1967  |       7.8100  |
  Worst of Gen.  |             80.0000  |         0.0123  |           75  |           10  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0583  |              35.0000  |
Evaluation Time  |             10.0200  |            6012.0000  |
================================================================================================


-----------------------
|-=    Generation 4 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |           35  |            8  |
Generation Mean  |             68.8850  |         0.0145  |      60.5167  |       7.6067  |
  Worst of Gen.  |             80.0000  |         0.0123  |           95  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0300  |              18.0000  |
Evaluation Time  |             10.4750  |            6285.0000  |
================================================================================================
```

Como se ha podido observar en el ejemplo proporcionado, el fichero se compone de dos elementos de información, por un lado la información de la generación y, por otro lado, información del mejor individuo si en esta generación se ha obtenido un nuevo mejor individuo.

Para cada generación se ofrece la siguiente información:

```
----------------------
|-=    Generation 0 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |              65.0000  |         0.0152  |           63  |            7  |
Generation Mean  |              79.6183  |         0.0124  |      67.0733  |       7.9100  |
  Worst of Gen.  |              80.0000  |         0.0123  |           77  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |               0.0000  |                 0.0000  |
Evaluation Time  |               5.4633  |              3278.0000  |
-------------------------------------------------------------------
```

En primer lugar el número de la generación en la que nos encontramos. Para cada generación se muestran dos bloques de información, por un lado información de los individuos que forman la generación y, por otro lado, el tiempo empleado en las operaciones en milisegundos.

El bloque principal es el que muestra información de los individuos, que nos indica el fitness puro y el fitness ajustado, así como el número de nodos y la profundidad máxima del árbol, del mejor y peor individuo de la generación y la media de ésta última.

En el segundo bloque se muestra, a título informativo, los tiempos empleados en la evolución y evaluación de la generación.

Esta información nos permite conocer el estado de las generaciones de un vistazo rápido, viendo su evolución a lo largo del tiempo.

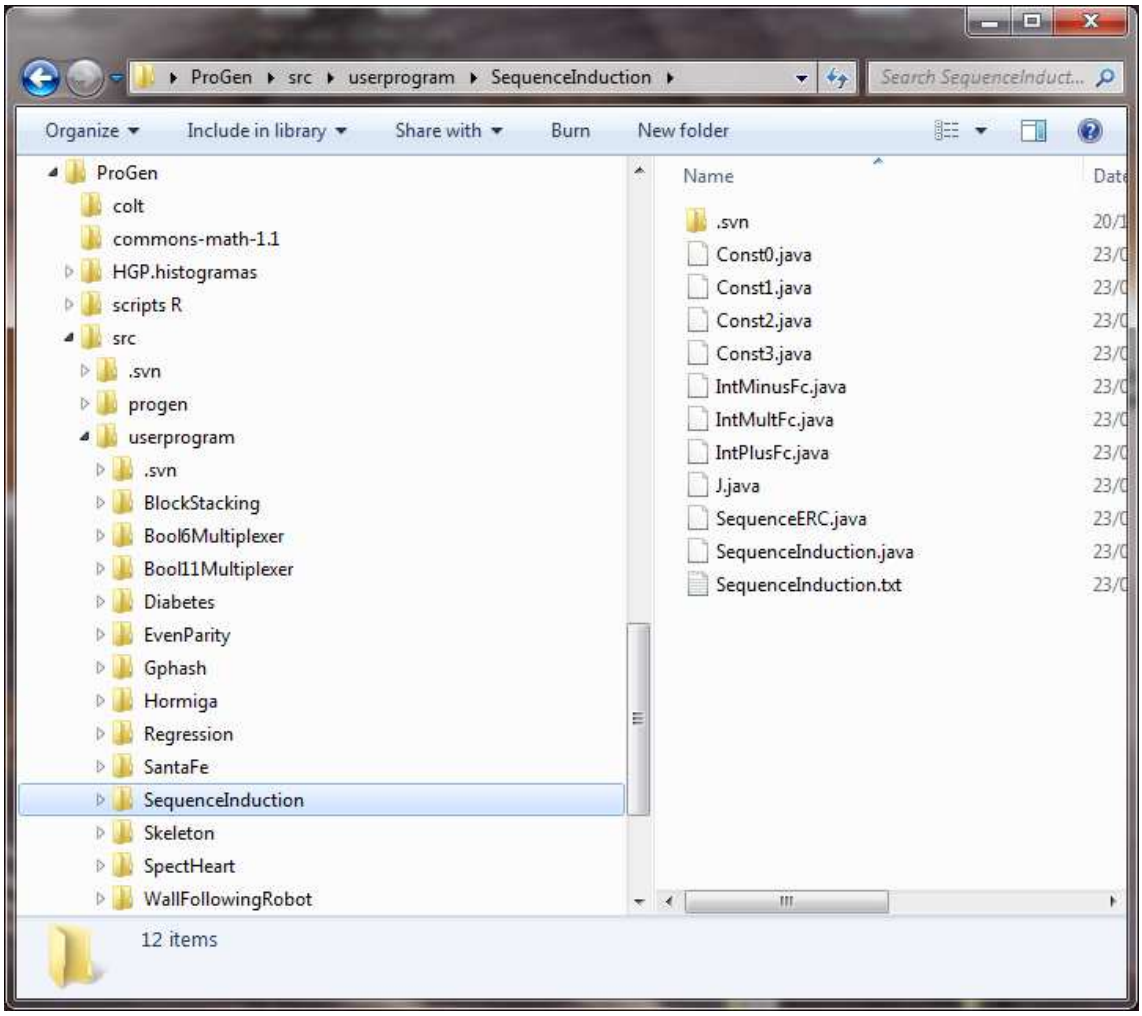
```
 ----------------------
|-=    Generation 3 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |          35  |           8  |
Generation Mean  |             73.8650  |         0.0135  |     69.1967  |      7.8100  |
  Worst of Gen.  |             80.0000  |         0.0123  |          75  |          10  |
------------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0583  |               35.0000  |
Evaluation Time  |             10.0200  |             6012.0000  |
================================================================================================


----------------------
|-=    Generation 4 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             55.0000  |         0.0179  |          35  |           8  |
Generation Mean  |             68.8850  |         0.0145  |     60.5167  |      7.6067  |
  Worst of Gen.  |             80.0000  |         0.0123  |          95  |           9  |
------------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0300  |               18.0000  |
Evaluation Time  |             10.4750  |             6285.0000  |
================================================================================================
```

Este es un caso en el que se puede contemplar el avance de las generaciones. Como se puede observar, en estas generaciones no se han obtenido nuevos mejores individuos, sin embargo la media de las generaciones va mejorando, lo que posibilitará la aparición de nuevos mejores individuos más adelante.

La información que se muestra de los nuevos mejores individuos, es la misma que se describió para el fichero best_individual.txt, descrito anteriormente.

2.3.2. Definición de problemas

Los problemas se desarrollan integrados dentro del propio ProGen, en la siguiente estructura de carpetas:



Dentro de la carpeta, o paquete, userprogram, se encuentran todos los problemas programados para ProGen. Dentro del paquete de cada problema, se encuentran los archivos fuente del problema, donde se encuentran los archivos fuente de los nodos, tanto operadores como terminales. También se encuentra en el mismo paquete1 la lógica del problema encargada de implementar las funciones específicas del problema, como su función de fitness y las acciones a llevar a cabo al inicializar el problema o al finalizarlo.

Además de lógica, se puede encontrar un archivo de configuración para el problema, en la imagen “SequenceInduction.txt”.

En este archivo se definen las propiedades del problema, como, por ejemplo, el archivo con la lógica del problema, el número de generaciones máximo para resolver el problema, el tamaño de la población, etc. El archivo es de la siguiente forma:

```
#*********************************************
#        Experiment properties file           *
#*********************************************


#----PROJECT----
#The project must be written in a java file with the same name that you specify in this property.
#it will be also used to name output files.
prp_project_name: SequenceInduction.SequenceInduction
#Available: English, Espaol
prp_language: espaol
#Output: detailed or summarized
output_mode: detailed
prp_hypergp: on
prp_hgp_generations: 5
prp_hgp_convergence_limit: 80%

#----POPULATION----
#You can load the initial population from an XML file if you dont want it to be generated randomly
prp_load_population: population.xml
prp_population_size: 200
prp_random_seed: 3
#Full, grow, half and half
#prp_initialization_mode: half and half
prp_initialization_mode: grow
#Maximun number of attempts to generate a valid tree (both for the initial population and during the evolution)
prp_max_attempts: 50

#----INDIVIDUALS----
#valid trees during the evolution
prp_max_nodes: 100
prp_max_depth: 10
#Valid depth range for the initial population trees
prp_depth_interval: 2,7

#----EVOLUTION----
prp_generations: 100
#Evolution will stop when this value (or any other inside the allowed error interval) is reached
#prp_stop_fitness: 0.0
#Upper and lower bounds from the stop_fitness value. Reaching a value in this interval will also stop the
evolution
#prp_error_interval: 0.0, 100.0
#checkpoints. Valid values are either a list of integers or/and the word "every" followed by an integer.
prp_checkpoints: 10, 47, 82, every 100

#exact number or percentage value
prp_elitism: 10%
prp_GP_operator_number: 2
prp_GP_op1_name: OnePointCrossover
prp_GP_op1_probability: 0.80
prp_GP_op1_selection: Tournament(size=10)

prp_GP_op2_name: Reproduction
prp_GP_op2_probability: 0.20
prp_GP_op2_selection: Roulette

#prp_GP_op3_name: Crossover(internal=0.9)
#prp_GP_op3_probability: 0.10
#prp_GP_op3_selection: Roulette
```

2.3.3. Módulo de programación genética en árbol

El rendimiento de la programación genética está muy relacionado con la calidad de la población inicial. Si la población inicial es suficientemente buena, los resultados serán muy buenos. Si la población inicial es muy mala, los resultados serán malos.

Por este motivo, a ProGen se le ha añadido un nuevo módulo cuya función no es resolver un problema, sino proporcionar poblaciones iniciales optimizadas a la programación genética.

Como ya se ha comentado anteriormente, la programación genética en árbol trata de evolucionar las formas de los árboles durante un número de generaciones, denominadas generaciones arbóreas para distinguirlas de las de la programación genética tradicional. Para cada problema se definen una serie de parámetros que se utilizarán durante la ejecución:

```
                #*******************************************
                #        Experiment properties file         *
                #*******************************************


#----PROJECT----
...
prp_hypergp: on
prp_hgp_generations: 5
prp_hgp_convergence_limit: 80%

...
```

Con estos tres parámetros se configura la programación genética en árbol. El primer parámetro permite activarla o desactivarla.

El parámetro prp_hgp_generations indica el número máximo de generaciones que se ejecutarán. Una vez alcanzada dicha generación, se procederá a la creación de la población inicial de la programación genética con el mejor individuo de los disponibles.

El parámetro prp_hgp_convergence_limit es otra condición de finalización. Indica el porcentaje de árboles iguales en la población que debe haber para detener la ejecución de generaciones arbóreas. De esta forma, cuando el 80% de los individuos de la generación arbórea tengan el mismo árbol, se procederá a la generación de la población tradicional.

Tras la inserción del módulo de programación genética, la secuencia de ejecución quedaría de la siguiente forma:

- Programación genética en árbol (opcional).
  - Generación de la población inicial de individuos arbóreos.
  - Evaluación de los individuos arbóreos.
  - Evolución de los individuos arbóreos.
  - Detección de fin de la programación genética en árbol.

- Programación genética
  - Generación de la población inicial de individuos tradicionales
  - Evaluación de los individuos
  - Evolución de los individuos
  - Detección del fin de la programación genética.

3. Ejecuciones

A continuación se describirán las ejecuciones que se han realizado con el motor de programación genética ProGen. Los experimentos se han realizado en igualdad de condiciones tanto para la programación genética como para la programación genética en árbol, lo que permitirá extraer conclusiones entre ambas.

Se han considerado condiciones de igualdad aquellas en las que se ejecutan los problemas con los mismos parámetros de ejecución, como puedan ser el tamaño de la población, los operadores genéticos ejecutados y el número de generaciones de programación genética. La única diferencia representativa es la ejecución previa de un determinado número de generaciones de programación genética en árbol en los casos en los que ésta esté activa.

Otro aspecto a tener en cuenta es que para cada problema se han ejecutado un total de 10 repeticiones. De esta forma los resultados no tendrán tanto en cuenta lo buena que haya sido una ejecución, sino lo buena que es, en término medio, una configuración para resolver el problema. Esto nos permitirá comparar cómo se comporta el motor si la programación genética en árbol está activada o no.

Inicialmente se habían programado 30 repeticiones en lugar de 10, pero el coste en tiempo no era asumible.

3.1. Problemas

3.1.1. Wall Following Robot

El problema del wall following robot consiste en conseguir, mediante programación genética, un programa que permita a un robot recorrer el perímetro de las paredes de una habitación.

Para la ejecución de este problema, se realiza una simulación de cada problema generado. En dicha simulación, se utiliza un robot que recorre una habitación.

El robot tiene las siguientes características:

- 12 sensores repartidos circularmente que miden las distancias a las paredes de la habitación.
- Motor que le permite avanzar o girar.

La habitación se compone de un conjunto de paredes a lo largo de las cuales se distribuyen checkpoints.

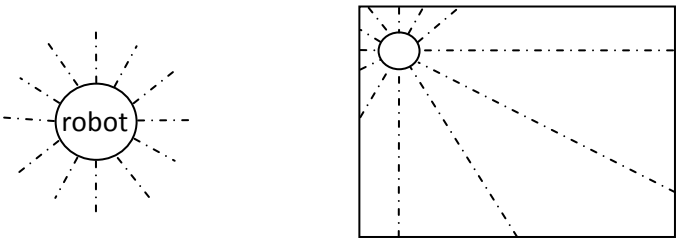


Ilustración 23 - Distribución de los sensores en el robot

El robot es capaz de obtener en todo momento las distancias que mide cada sensor. Esto se realiza mediante la resolución de ecuaciones matemáticas.

Al motor de programación genética se le dan un conjunto de herramientas para manejar el robot. Las herramientas son:

- Obtener el valor de uno de los doce sensores.
- Una herramienta condicional sí.
- Movimiento hacia delante y hacia detrás
- Giro a la derecha y a la izquierda.

Para evaluar los individuos, se reparten entre todas las paredes un conjunto de checkpoints de tal forma que, cada vez que el robot pasa por un checkpoint lo marca, y finalmente el individuo que haya marcado todos los checkpoints será el que haya recorrido las paredes.

La implementación de este problema, se ha definido conforme a la definición de este mismo problema que propuso John R. Koza, que se puede observar en el Anexo 1.

Inicialmente se tomó como medida de la habitación en la que se resolvía el problema un modelo de 4 x 4, resultando este problema trivial, siendo resuelto en muy pocas iteraciones, por lo que finalmente se decidió ampliar el tamaño de las paredes a 10 unidades cada una, formando un perímetro total de 40 unidades. Finalmente se decidió que el número de checkpoints a colocar a lo largo del perímetro de la pared fuese de 80, correspondiendo a 2 checkpoints a cada unidad de longitud de la pared. Resultando finalmente de la siguiente forma:

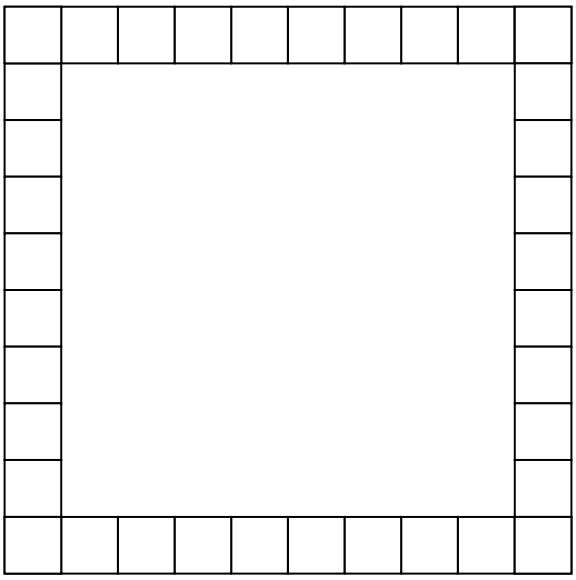


Ilustración 24 - Distribución de checkpoints en la habitación

3.2. Experimento 1

Para el primer experimento se ha realizado una ejecución del problema WallFollowingRobot con la siguiente configuración:

```
#Output: detailed or summarized
output_mode: detailed
prp_hypergp: on
prp_hgp_generations: 100
prp_hgp_convergence_limit: 95%
prp_hgp_sample_size: 5
prp_hgp_reevaluation_sample_size = 1

#----POPULATION----
#You can load the initial population from an XML file if you dont want it to be generated randomly
prp_load_population: population.xml
prp_population_size: 600
prp_random_seed: 1234567890
#Full, grow, half and half
prp_initialization_mode: grow
#Valid depth range for the initial population trees
prp_depth_interval: 4, 10
#Maximun number of attempts to generate a valid tree (both for the initial population and during the evolution)
prp_max_attempts: 1000

#----INDIVIDUALS----
#valid trees during the evolution
prp_max_nodes: 100
prp_max_depth: 30

#----FUNCTIONS and ADF'S----
#if you want to use ADF'S name them like ADF0, ADF1 and so. To refer its branches use ARG0, ARG1...
prp_num_function_sets: 1
prp_function_set_0: Prog2Fc, IfLwEqualsThanFc, S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, SS, MoveForeward,
MoveBackward, TurnLeft, TurnRight
prp_return_type_fs_0: void
prp_function_set_1: AntMove, AntIfFoodAheadFc, Prog2Fc, ARG0
prp_return_type_fs_1: void
prp_number_of_trees: 1
prp_tree0_function_set_number: 0
prp_number_of_adfs: 0
prp_ADF0_function_set_number: 1
prp_ADF0_interface: void$$void

#----EVOLUTION----
prp_generations: 200
#Evolution will stop when this value (or any other inside the allowed error interval) is reached
#prp_stop_fitness: 0
#Upper and lower bounds from the stop_fitness value. Reaching a value in this interval will also stop the evolution
#prp_error_interval: 5, 0
#checkpoints. Valid values are either a list of integers or/and the word "every" followed by an integer.
prp_checkpoints: 10, 47, 82, every 100
#exact number or percentage value
prp_elitism: 2%
```

```
#Genetic operators to be applied
prp_GP_operator_number: 3
prp_GP_op1_name: OnePointCrossover(internal=0.8)
prp_GP_op1_probability: 0.05
prp_GP_op1_selection: Tournament(size=4)

prp_GP_op2_name: Reproduction(internal=1.0)
prp_GP_op2_probability: 0.05
prp_GP_op2_selection: Roulette

prp_GP_op3_name: PointMutation(internal=1)
prp_GP_op3_probability: 0.90
prp_GP_op3_selection: Tournament(size=4)

prp_HGP_operator_number: 2
prp_HGP_op1_name: Crossover(internal=0.8)
prp_HGP_op1_probability: 0.90
prp_HGP_op1_selection: Tournament(size=10)

prp_HGP_op2_name: Reproduction(internal=1.0)
prp_HGP_op2_probability: 0.10
prp_HGP_op2_selection: Roulette
```

Y la siguiente configuración para el experimenter:

```
#EXPERIMENTER
prp_experimenter: on
prp_experimenter_repetitions: 10
prp_hypergp: on, off
prp_generations: 50,200:50
prp_hgp_generations: 10,160:50
```

La configuración descrita anteriormente se traduce en las siguientes condiciones:

50 generaciones de programación genética.
600 individuos en cada población.
100 nodos como máximo para cada individuo de la población.
30 niveles de profundidad como máximo para cada individuo de la población.
10 Repeticiones para cada configuración, finalmente se tomara el fitness medio.
La mitad de las ejecuciones tendrán activada la PGA y la otra mitad no.
De 50 a 200 generaciones, incrementándolas de 50 en 50.
De 10 a 160 generaciones arbóreas, incrementándolas de 50 en 50.

3.2.1. Resultados

Según la configuración con la que se ha realizado la ejecución, se puede realizar una comparación sobre las ejecuciones que tenían activada la programación genética en árbol y las que no. En esta comparación se han observado los siguientes resultados:

Ejecución con PGA

Experimento	Average Fitness	Best Fitness
exp_0	52,000	44,000
exp_2	53,800	45,000
exp_4	54,600	46,000
exp_6	57,600	49,000
exp_8	44,800	41,000
exp_10	46,500	35,000
exp_12	45,900	42,000
exp_14	44,200	37,000
exp_16	44,700	38,000
exp_18	43,100	37,000
exp_20	43,100	32,000
exp_22	45,000	41,000
exp_24	43,100	37,000
exp_26	46,200	36,000
exp_28	46,400	43,000
exp_30	45,200	40,000

Ejecución sin PGA

Experimento	Average Fitness	Best Fitness
exp_1	62,900	57,000
exp_3	62,700	58,000
exp_5	64,200	57,000
exp_7	63,100	61,000
exp_9	62,700	60,000
exp_11	62,300	51,000
exp_13	63,400	58,000
exp_15	63,200	52,000
exp_17	64,000	58,000
exp_19	62,900	53,000
exp_21	59,600	44,000
exp_23	63,400	61,000
exp_25	62,600	59,000
exp_27	64,100	57,000
exp_29	61,900	58,000
exp_31	62,100	59,000

Tabla 1 - Resultados por tipo de programación genética del experimento 1

Como se puede observar, la ejecución con la programación genética en árbol activa ha obtenido mejores resultados en todos los experimentos; tan sólo el experimento 21, que no la tenía activada, ha conseguido (en la mejor de sus ejecuciones) un fitness que podría competir con los fitness obtenidos en la ejecución con PGA.

A continuación se ofrece un gráfico que ofrece visualmente la diferencia entre las ejecuciones:

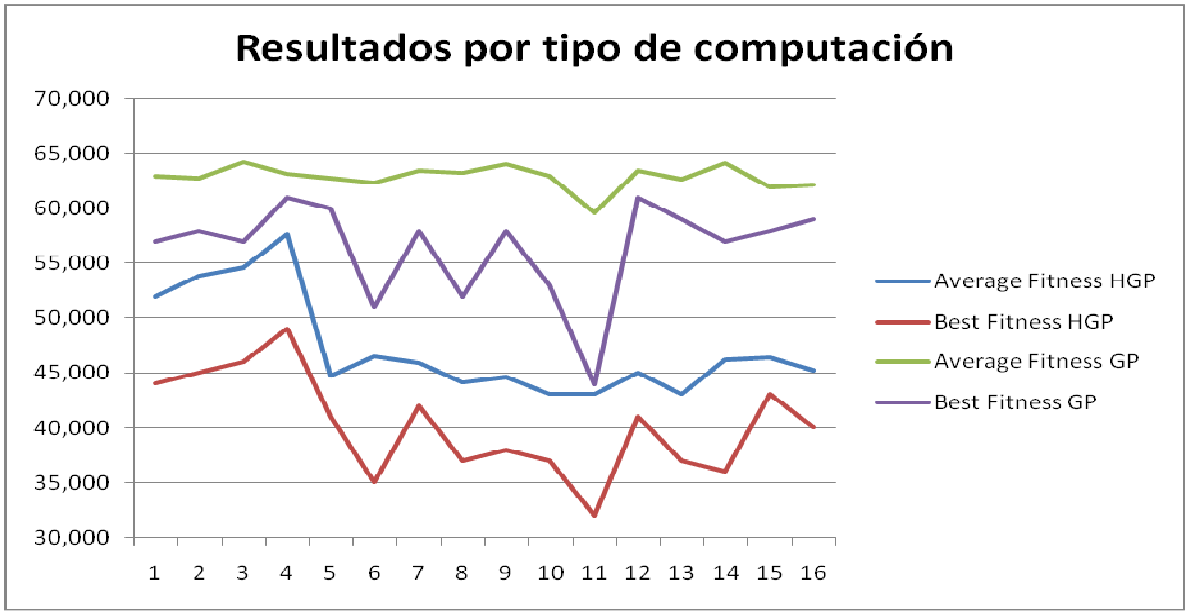


Tabla 2 - Gráfica de resultados por tipo de programación del experimento 1

Como se ha comentado, las ejecuciones con la programación genética en árbol activada han obtenido unos mejores resultados tanto absolutos, como en media. Se puede comprobar cómo el caso observado en las tablas representa un pico óptimo en Best Fitness GP, pero el comportamiento es peor. También se puede observar cómo las líneas medias están más suavizadas que las del mejor individuo, con lo que ofrecen un resultado más fiable.

Otra comprobación que se quería abordar en este experimento es evaluar el rendimiento de distinto número de ejecuciones de programación genética en árbol, puesto que en el experimento, las ejecuciones se han realizado variando el número de generaciones arbóreas entre 10, 60, 110 y 160. Estas ejecuciones nos han ofrecido los siguientes resultados:

Ejecución con 10 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_0	52,000	44,000
exp_2	53,800	45,000
exp_4	54,600	46,000
exp_6	57,600	49,000

Ejecución con 60 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_8	44,800	41,000
exp_10	46,500	35,000
exp_12	45,900	42,000
exp_14	44,200	37,000

Ejecución con 110 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_16	44,700	38,000
exp_18	43,100	37,000
exp_20	43,100	32,000
exp_22	45,000	41,000

Ejecución con 160 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_24	43,100	37,000
exp_26	46,200	36,000
exp_28	46,400	43,000
exp_30	45,200	40,000

Tabla 3 - Resultados por número de generaciones arbóreas

A partir de estas tablas, se puede construir los siguientes gráficos:

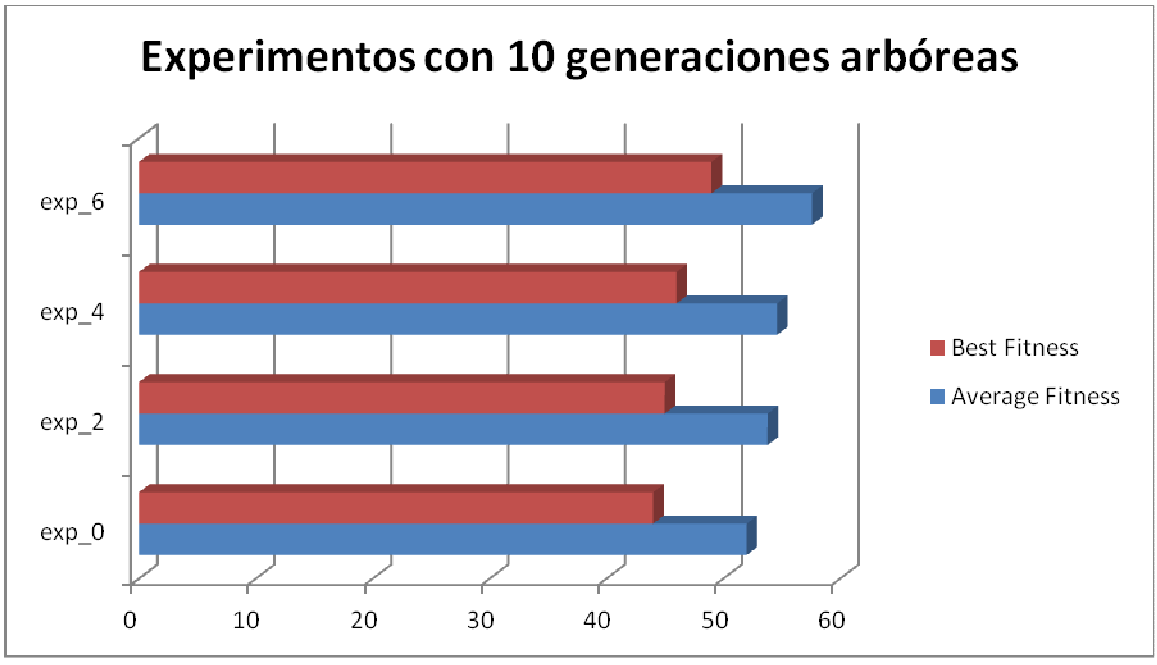


Tabla 4 - Gráfica de resultados con 10 generaciones arbóreas del experimento 1

En esta gráfica se puede observar el comportamiento de distintos experimentos habiendo fijado el número de generaciones arbóreas en 10. Los experimentos que cumplen esta característica (con la programación genética en árbol activa) son el 0, 2, 4 y 6. Las diferencias entre estos experimentos son el número de generaciones de programación genética que se han utilizado durante su evolución, siendo esta 50, 100, 150 y 200 generaciones.

Paradójicamente, el experimento que mejores resultados da es el experimento 0, que tan sólo ha contado con 50 generaciones de programación genética, mientras que según se ha ido incrementando dicho número, los resultados empeoran. De esto se deduce que superado el tiempo adaptación, ya no sólo no consigue mejorar, sino que empeora sensiblemente.

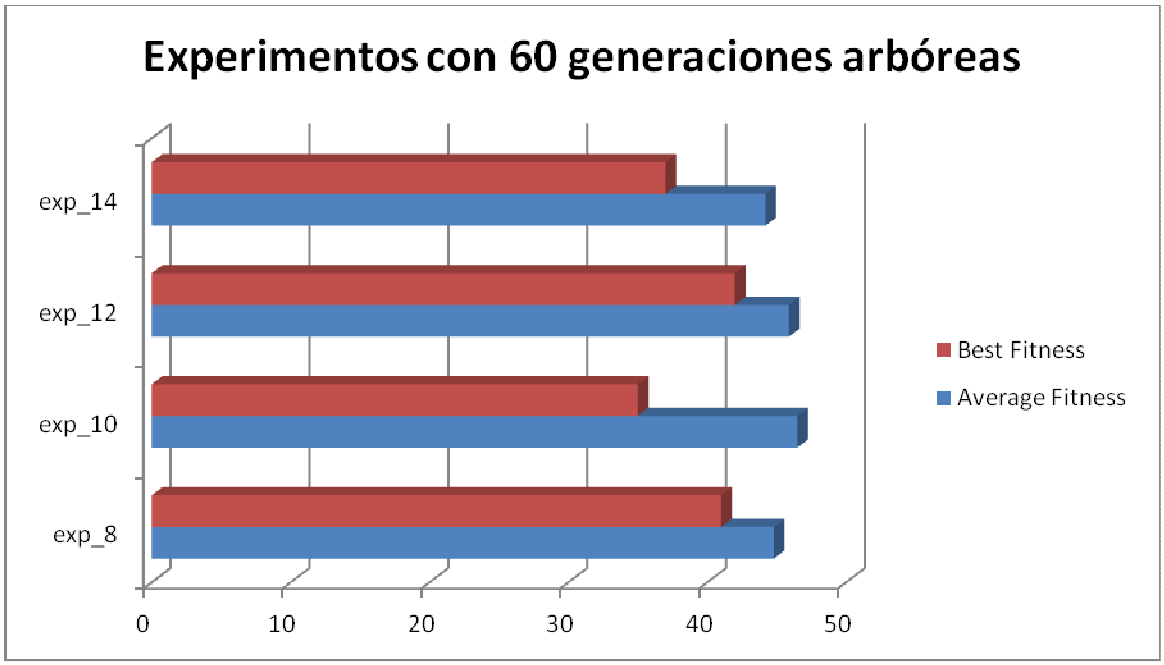


Tabla 5 - de resultados con 60 generaciones arbóreas del experimento 1

En la gráfica superior se puede observar los resultados obtenidos por los experimentos que tenían fijado el número de generaciones arbóreas a 60. En esta ocasión se puede observar que los resultados son mejores que los de la ejecución anterior, estando todos ellos por debajo de 50.

Aunque en esta ocasión, la ejecución con 50 generaciones de programación genética (experimento 8) ha obtenido también unos buenos resultados, se puede observar un comportamiento más previsible en los otros 3 experimentos, en los que se puede observar que con un mayor número de generaciones se va obteniendo un mejor resultado, llegando a 44,2 en el experimento 14.

En el caso del experimento 10, se puede observar como al haber ejecutado 10 repeticiones para cada experimento, nos permite filtrar las ejecuciones que aleatoriamente dan resultados extremos, como se puede observar en el Best Fitness que llega a 35, el mejor resultado de los cuatro experimentos.

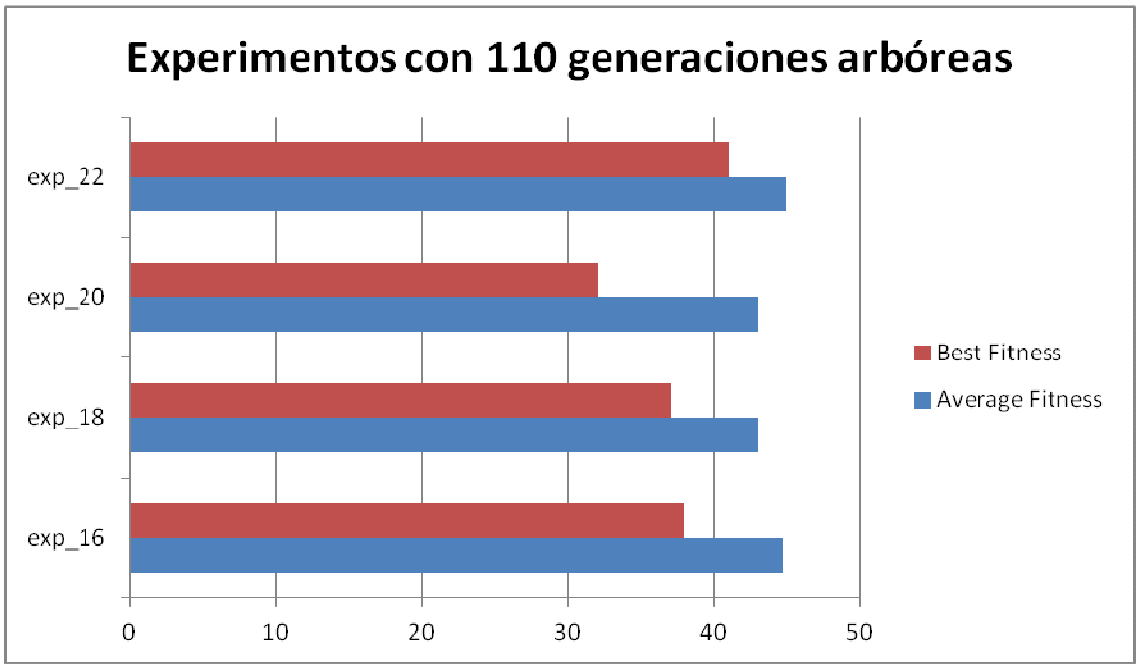


Tabla 6 - de resultados con 110 generaciones arbóreas del experimento 1

En esta gráfica se ofrecen los resultados para los experimentos con el número de generaciones arbóreas establecido a 110. A diferencia de los casos anteriores, esta vez los mejores resultados se observan en los experimentos intermedios, que se corresponden con 100 y 150 generaciones.

En este caso se ha conseguido corregir el mínimo del experimento 20, que habría vuelto a desviar los resultados. En apariencia el número de generaciones utilizado después de la programación genética en árbol no es relevante, no se está observando una mejora directa sobre los resultados al disponer de un mayor número de generaciones para mejorar.

Por último observaremos los resultados correspondientes a 160 generaciones arbóreas.

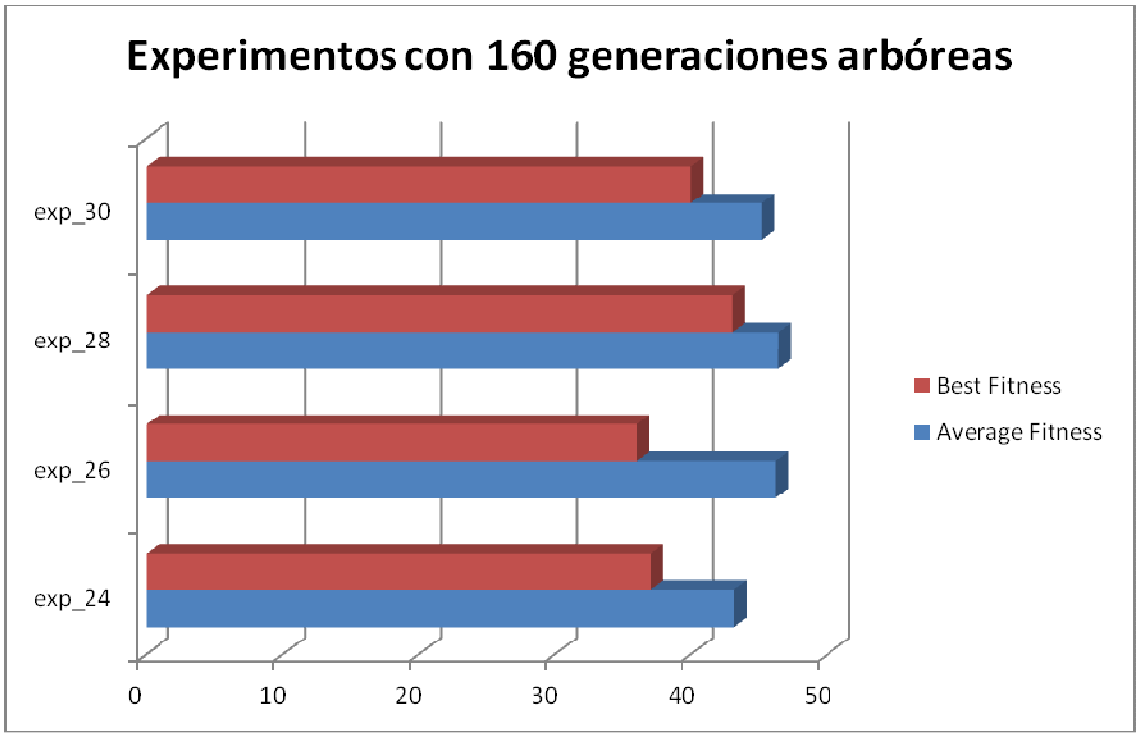


Tabla 7 - de resultados con 160 generaciones arbóreas del experimento 1

En este experimento los mejores resultados los vuelven a dar aquellos con 50 y 200 generaciones de programación genética, en contraposición directa con los resultados del valor anterior.

Finalmente vamos a obtener una gráfica en la que se puedan observar los valores medios de todos los experimentos, representando los valores según el número de generaciones arbóreas y en función del número de generaciones.

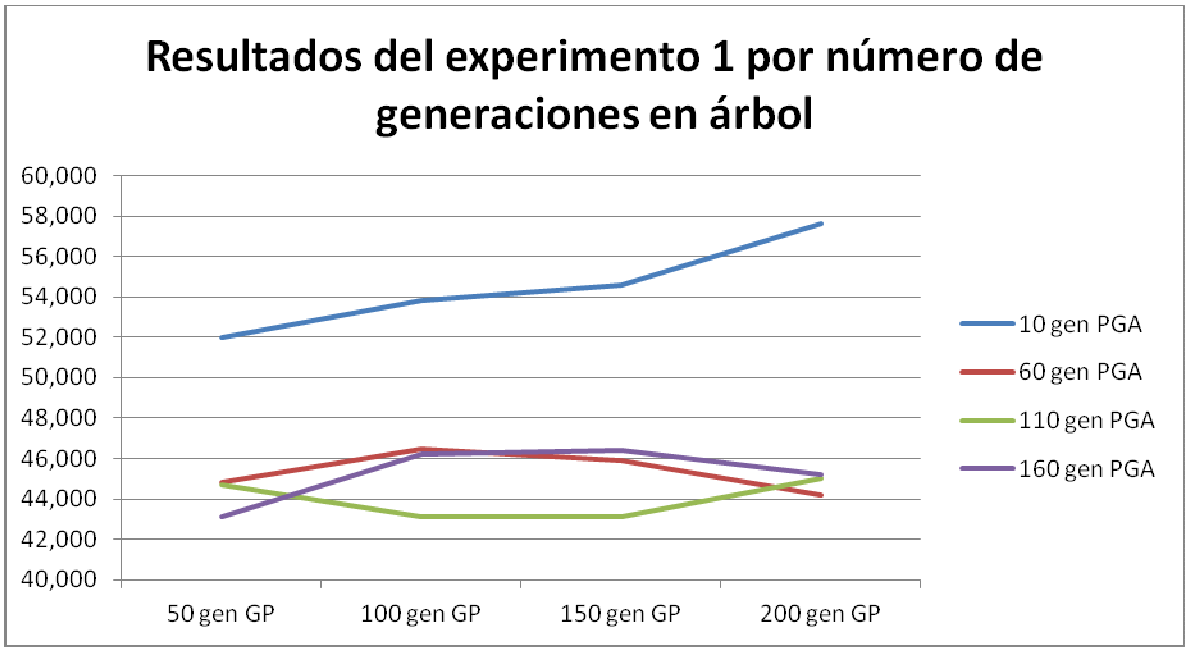


Tabla 8 - Gráfica de resultados por número de generaciones arbóreas del experimento 1

En este gráfico puede observarse un dato bastante interesante: 10 generaciones de programación genética en árbol aparentan ser un número insuficiente, y ofrece peores resultados que los otros tres números de generaciones.

Para el resto de las ejecuciones se puede observar que una vez configuradas un número determinado de generaciones arbóreas, el número de generaciones de programación genética no es representativo, o lo que es lo mismo, habiendo utilizado la programación genética en árbol, los resultados están optimizados y no hacen falta muchas generaciones de programación genética.

El último dato a comparar en esta ejecución es el rendimiento de la programación genética. En este caso, la configuración del problema era claramente inadecuada a la programación genética tradicional.

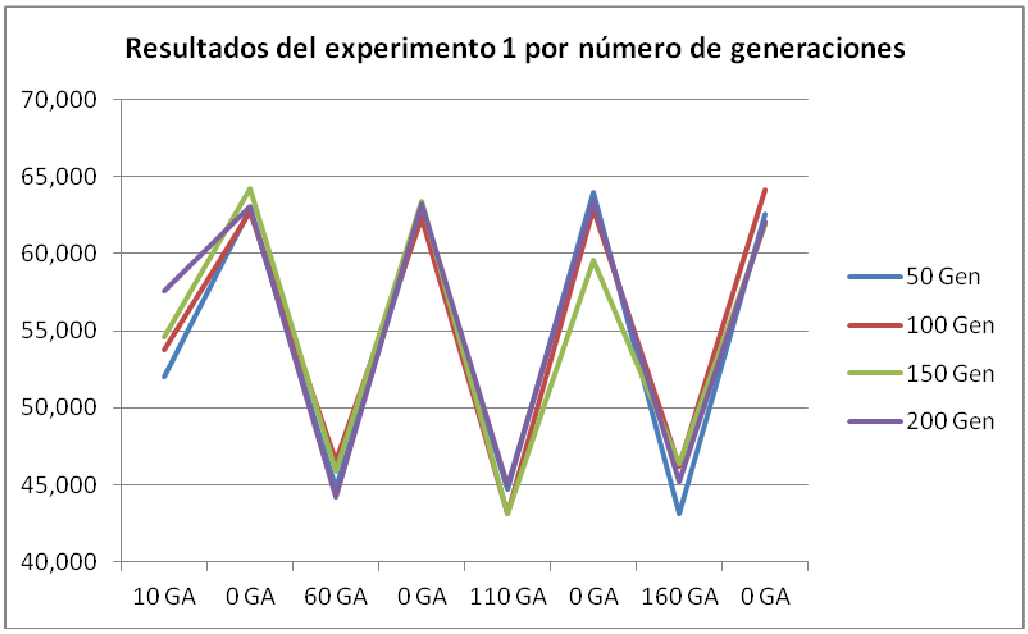


Tabla 9 - Gráfica de resultados por número de generaciones del experimento 1

Como se puede comprobar en el gráfico, el comportamiento es el mismo, independientemente de que el número de generaciones sea 50 o 200. En este caso no aporta una mejor solución. Dada la naturaleza del experimenter, que procesa las combinaciones de experimentos secuencialmente, se puede observar que las ejecuciones impares son los que tenían la programación genética en árbol activa, mientras que los pares no. Del mismo modo se puede observar que la primera ejecución es la que tan sólo hacía 10 generaciones arbóreas, obteniendo peores resultados que los otros 4 valores.

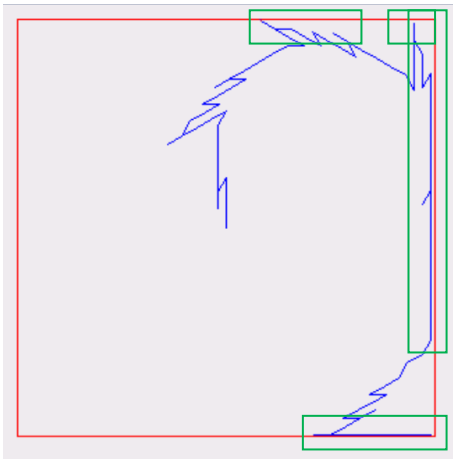
Hasta ahora hemos visto las estadísticas de los resultados, ahora podremos observar los resultados obtenidos mediante una simulación de un recorrido sobre la habitación de prueba. Se han tomado como ejemplo las mejores y peores ejecuciones.

Se debe tener en cuenta que los casos que han dado peores resultados, es en realidad el mejor peor resultado, porque de cada resultado se ha realizado una batería de 10 repeticiones y de esas 10 repeticiones, el resultado mostrado es el mejor de todos ellos.

Teniendo esto en cuenta, también se ha sacado el peor resultado atendiendo a las distintas repeticiones, quedando nombrados las imágenes de la forma “E1 Mejor Mejor robot con PGA”, queriendo indicar que para el experimento 1 se muestra el resultado de la mejor repetición de la mejor batería con la programación genética en árbol activada.

3.2.2. Visualización de los resultados

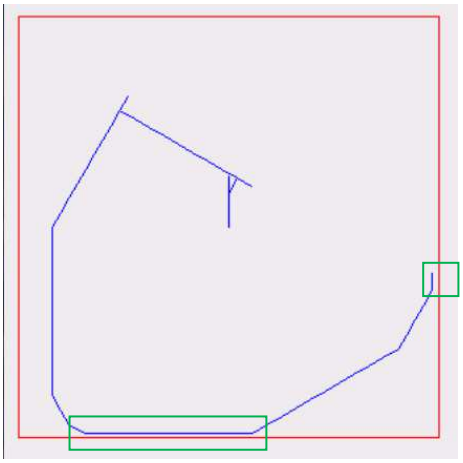
Mejor resultado con la programación genética en árbol activa:



E1 Mejor Mejor Robot con HGP.avi

Ilustración 25- Mejor individuo del experimento 1 con PGA

Mejor resultado con la programación genética en árbol desactivada:



E1 Mejor Mejor Robot sin HGP.avi

Ilustración 26 - Mejor individuo del experimento 1 sin PGA

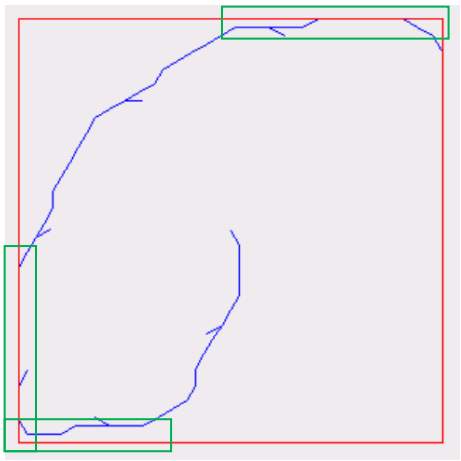
Como se puede observar, ambos robots son capaces de avanzar desde el centro de la habitación hasta las paredes y adaptarse a la dimensión de la misma.

La solución a este problema no es trivial, puesto según definió el problema Koza, sólo puntúa un individuo una vez ha llegado a la pared, mientras que se está moviendo por el interior de la habitación no es capaz de detectar mejoras.

Para resolver esto se podría haber planteado una especie de campo de potenciales con las distancias a las paredes, pero eso queda fuera de la definición de Koza.

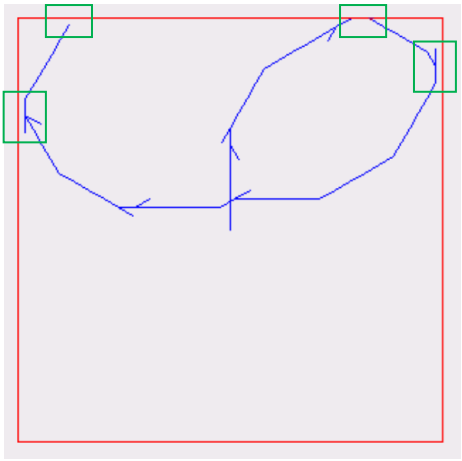
En este ejemplo se puede comprobar de forma visual lo que hasta ahora habían demostrado los números, los resultados con programación genética en árbol son mejores y ofrecen una solución mejor adaptada al problema.

Esto también se observa en los peores resultados:



E1 Mejor Peor Robot con HGP.avi

Ilustración 27 - Peor individuo de entre las baterías de prueba del experimento 1 con PGA



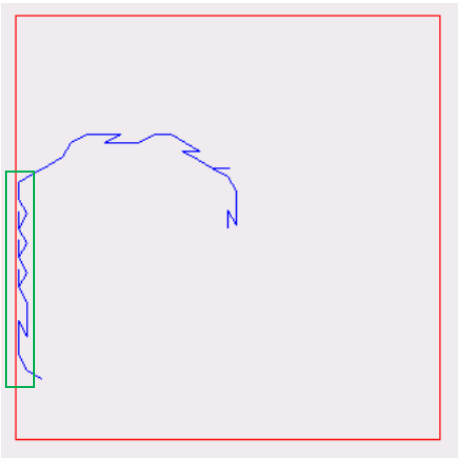
E1 Mejor Peor Robot sin HGP.avi

Ilustración 28 - Peor individuo de las baterías de prueba el experimento 1 sin PGA

En estos resultados se puede observar un cierto grado de adecuación a la solución, los robots son capaces de girar con la habitación y avanzar más o menos recto. Esto se debe a que como se ha mencionado anteriormente, las soluciones mostradas son los peores

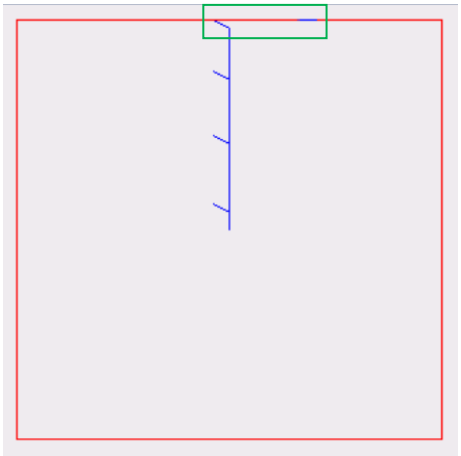
resultados de entre todas las baterías de pruebas, pero hay que tener en cuenta que para cada batería de pruebas se han realizado 10 repeticiones. Por lo tanto, estos son los mejores de entre las 10 repeticiones.

A continuación mostraremos los que realmente han sido los peores resultados de todas las repeticiones.



E1 Peor Peor Robot con HGP.avi

Ilustración 29 - Peor individuo de las repeticiones del experimento 1 con PGA



E1 Peor Peor Robot sin HGP.avi

Ilustración 30 - Peor individuo de las repeticiones del experimento 1 sin PGA

Con estos últimos resultados se puede observar que realmente el comportamiento de algunas soluciones planteadas puede llegar a ser muy malo, aunque en apariencia tenga un comportamiento similar al que pudiese pensar un ser humano. Este es el caso de la

última solución, en la que el robot avanza recto hasta la pared, y gira para seguir avanzando por ella.

El problema de esta solución, en realidad la peor de todas, es que el individuo avanza mucho menos que con cualquiera de las otras soluciones, siendo por ello, las otras mejores.

3.2.3. Conclusiones

En las ejecuciones de este experimento se han obtenido unos resultados muy interesantes.

A pesar de que la configuración era totalmente opuesta a la programación genética, que no consiguió obtener ningún resultado, la programación genética en árbol, ha demostrado ser capaz de, si bien no obtener soluciones directas al problema, si facilitar el camino a las mismas y ofrecer a la programación genética una población inicial más próxima a la solución que una población aleatoria.

El experimento ha demostrado claramente la ventaja de utilizar programación genética en árbol, puesto que ofrece una mejora de fitness de casi 20 puntos en la mayoría de las ejecuciones, y respecto a esto hay que tener en cuenta que son promedios. Cada ejecución se ha repetido 10 veces para garantizar que no se ofrecen resultados que se desvíen demasiado de una ejecución “normal”.

Para este experimento se ha demostrado que el número de 10 generaciones arbóreas es un número insuficiente, mientras que los mejores resultados se han obtenido con 110 generaciones arbóreas, aunque las ejecuciones con 60 tampoco han sido malas. De esto se puede extraer que para este problema una oscilación entre 60 y 120 generaciones arbóreas debería ser un número adecuado para el experimenter.

Por último se quiere comentar, que aunque este primer experimento ha dejado en clara evidencia la conveniencia de utilizar la programación genética en árbol en determinadas situaciones, no nos permite comparar ambas programaciones debido a que la programación genética no ha conseguido funcionar casi en ningún caso. Por ello esta comparación se realizará con los resultados de otros experimentos más adecuados.

Respecto a los resultados gráficos, se han observado algunos resultados bastante interesantes.

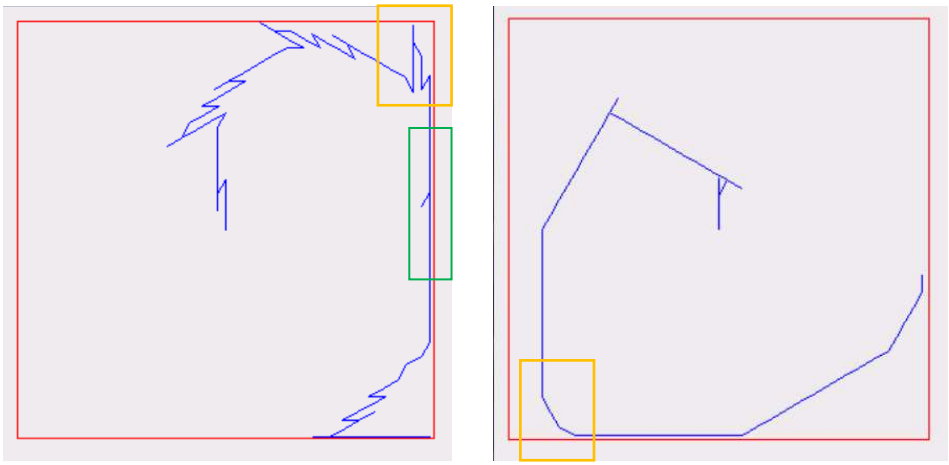


Ilustración 31 - Mejores resultados del experimento 1

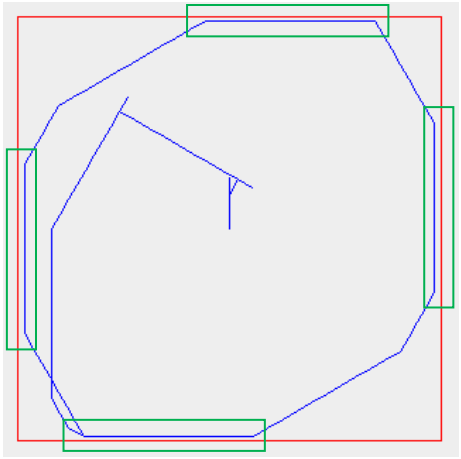
Se puede comprobar que el resultado del robot, ejecutándose con programación genética en árbol es mejor que el proporcionado por la programación genética tradicional. En ambos casos se puede observar que el robot es capaz de seguir una pared y rectificar en caso de haberse producido una desviación, como se puede observar en el fragmento recuadrado en verde.

También se observa en la segunda imagen que es capaz de realizar recorridos mucho más rectilíneos y directos que el primero.

Además, tenemos una adaptación, más o menos suave de los recorridos a las curvas de las paredes, como se observan en los recuadros amarillos.

En este sentido hay que tener en cuenta que los individuos de este problema, por la propia naturaleza de los terminales del problema, son muy complejos. Para hacer un recorrido recto es necesario encadenar múltiples operadores de avanzar o retroceder, sin que ningún giro aparezca en medio. Sería mucho más sencillo con un operador que avanzase hasta tocar la pared, pero eso realmente sólo sería útil en habitaciones bastante regulares. El hecho es que en realidad el programa que se evalúa y se visualiza se ejecuta varias veces de forma consecutiva, teniendo que adaptarse a ejecuciones desde varios puntos de partida. Es decir, al ver moverse un robot, no estamos viendo los operadores que componen un individuo una única vez, sino que cuando termina de ejecutarse un individuo este vuelve a ejecutarse sobre el robot, con la diferencia de que esta vez el robot ya no estará en el centro de la habitación sino en la posición en la que terminó la ejecución anterior.

A raíz de esto se puede observar algo muy interesante, porque si ejecutamos el mejor individuo de programación genética tradicional el doble de veces que estaba cuando se evaluó obtenemos el siguiente comportamiento:



E1 Mejor Mejor Robot sin HGP x2.avi

Ilustración 32 - Ejecución el doble de veces del mejor individuo sin PGA

Un comportamiento mucho más similar a lo que se podría esperar, que recorre buena parte de las cuatro paredes. De esto se puede concluir que los individuos serían capaces de proporcionar mejores resultados si la habitación fuese más pequeña o los individuos más grandes.

3.3. Experimento 2

Para este experimento se ha realizado una ejecución del problema WallFollowingRobot con la siguiente configuración:

```
#Output: detailed or summarized
output_mode: detailed
prp_hypergp: on
prp_hgp_generations: 100
prp_hgp_convergence_limit: 95%
prp_hgp_sample_size: 5
prp_hgp_reevaluation_sample_size = 1

#----POPULATION----
#You can load the initial population from an XML file if you dont want it to be generated randomly
prp_load_population: population.xml
prp_population_size: 600
prp_random_seed: 1234567890
#Full, grow, half and half
prp_initialization_mode: grow
#Valid depth range for the initial population trees
prp_depth_interval: 4, 10
#Maximun number of attempts to generate a valid tree (both for the initial population and during the evolution)
prp_max_attempts: 1000

#----INDIVIDUALS----
#valid trees during the evolution
prp_max_nodes: 100
prp_max_depth: 30

#----FUNCTIONS and ADF'S----

#if you want to use ADF'S name them like ADF0, ADF1 and so. To refer its branches use ARG0, ARG1...

prp_num_function_sets: 1
prp_function_set_0: Prog2Fc, IfLwEqualsThanFc, S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, SS, MoveForeward,
MoveBackward, TurnLeft, TurnRight
prp_return_type_fs_0: void
prp_function_set_1: AntMove, AntIfFoodAheadFc, Prog2Fc, ARG0
prp_return_type_fs_1: void
prp_number_of_trees: 1
prp_tree0_function_set_number: 0
prp_number_of_adfs: 0
prp_ADF0_function_set_number: 1
prp_ADF0_interface: void$$void

#----EVOLUTION----
prp_generations: 200
#Evolution will stop when this value (or any other inside the allowed error interval) is reached
#prp_stop_fitness: 0
#Upper and lower bounds from the stop_fitness value. Reaching a value in this interval will also stop the evolution
#prp_error_interval: 5, 0
#checkpoints. Valid values are either a list of integers or/and the word "every" followed by an integer.
prp_checkpoints: 10, 47, 82, every 100
#exact number or percentage value
prp_elitism: 2%
```

```
#Genetic operators to be applied
prp_GP_operator_number: 3
prp_GP_op1_name: OnePointCrossover(internal=0.8)
prp_GP_op1_probability: 0.55
prp_GP_op1_selection: Tournament(size=4)

prp_GP_op2_name: Reproduction(internal=1.0)
prp_GP_op2_probability: 0.05
prp_GP_op2_selection: Roulette

prp_GP_op3_name: PointMutation(internal=1)
prp_GP_op3_probability: 0.40
prp_GP_op3_selection: Tournament(size=4)

prp_HGP_operator_number: 3
prp_HGP_op1_name: Crossover(internal=0.8)
prp_HGP_op1_probability: 0.80
prp_HGP_op1_selection: Tournament(size=10)

prp_HGP_op2_name: Reproduction(internal=1.0)
prp_HGP_op2_probability: 0.10
prp_HGP_op2_selection: Roulette

prp_HGP_op3_name: OnePointCrossover(internal=0.8)
prp_HGP_op3_probability: 0.10
prp_HGP_op3_selection: Tournament(size=4)
```

Para el experimenter se ha utilizado esta otra configuración, que sobrescribe parte de la configuración anterior:

```
#EXPERIMENTER
prp_experimenter: on
prp_experimenter_repetitions: 10
prp_hypergp: on, off
prp_generations: 50,200:50
prp_hgp_generations: 10,160:50
```

La configuración descrita anteriormente se traduce en las siguientes condiciones:

50 generaciones de programación genética.
600 individuos en cada población.
100 nodos como máximo para cada individuo de la población.
30 niveles de profundidad como máximo para cada individuo de la población.
10 Repeticiones para cada configuración, finalmente se tomara el fitness medio.
La mitad de las ejecuciones tendrán activada la PGA y la otra mitad no.
De 10 a 160 generaciones, incrementando de 50 en 50, de programación genética en árbol.

Como se puede observar, las condiciones de ejecución son las mismas que en el experimento anterior, la diferencia viene expresada por los operadores genéticos utilizados, especialmente con la probabilidad de ejecutar cada una. En concreto se va a utilizar mucho más el operador OnePointCrossover (pasa de una probabilidad de 0.05 a 0.55) y se reduce mucho el uso de PointMutation (pasa de una probabilidad de 0.90 a 0.40).

Respecto a la programación genética en árbol, los cambios son muy reducidos, hemos reducido ligeramente el uso del operador Crossover, que pasa de una probabilidad de 0.90 a 0.80, y hemos incluido el uso del operador OnePointCrossover con una probabilidad de 0.10.

3.3.1. Resultados

Según la configuración con descrita anteriormente se han encontrado los siguientes resultados:

Ejecución con PGA

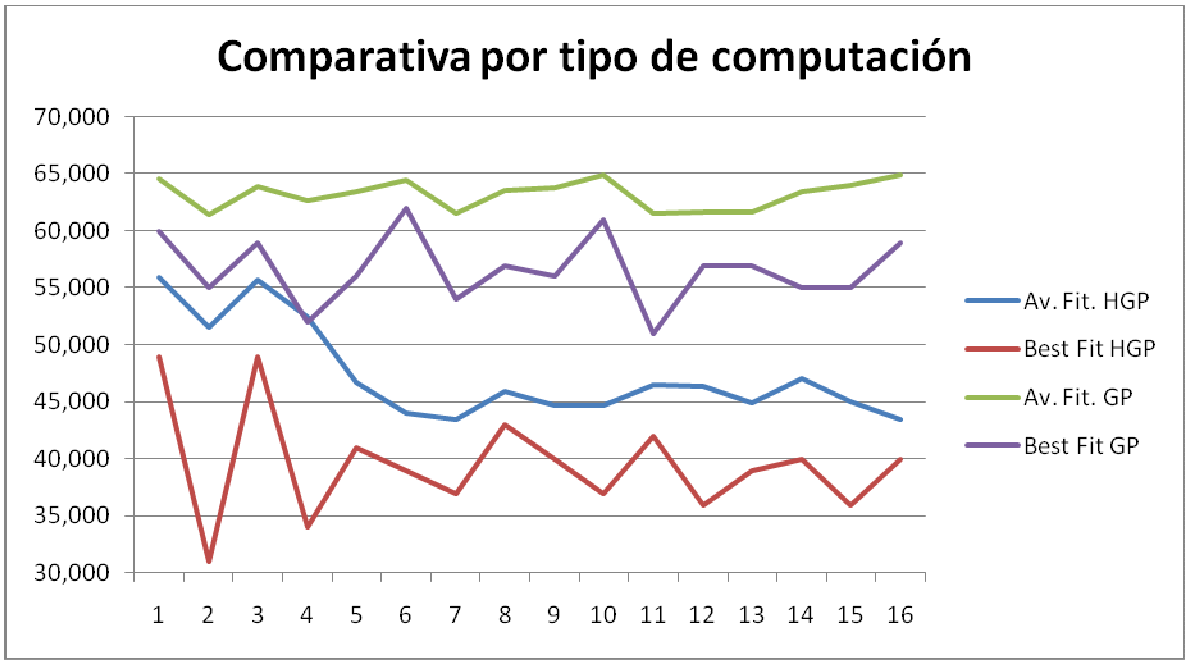
Experimento	Average Fitness	Best Fitness
exp_0	55,900	49,000
exp_2	51,500	31,000
exp_4	55,700	49,000
exp_6	52,400	34,000
exp_8	46,700	41,000
exp_10	44,100	39,000
exp_12	43,500	37,000
exp_14	46,000	43,000
exp_16	44,700	40,000
exp_18	44,700	37,000
exp_20	46,500	42,000
exp_22	46,400	36,000
exp_24	44,900	39,000
exp_26	47,100	40,000
exp_28	45,100	36,000
exp_30	43,500	40,000

Ejecución sin HGP

Experimento	Average Fitness	Best Fitness
exp_1	64,600	60,000
exp_3	61,400	55,000
exp_5	63,900	59,000
exp_7	62,600	52,000
exp_9	63,400	56,000
exp_11	64,500	62,000
exp_13	61,500	54,000
exp_15	63,600	57,000
exp_17	63,800	56,000
exp_19	64,900	61,000
exp_21	61,500	51,000
exp_23	61,600	57,000
exp_25	61,600	57,000
exp_27	63,400	55,000
exp_29	64,000	55,000
exp_31	64,900	59,000

En esta ejecución se puede observar cómo la programación genética en árbol ofrece unos mejores resultados que la simple programación genética en árbol. La mejor ejecución de la programación genética ofrece un fitness medio de 61,4, mientras que la ejecución de la programación genética en árbol ofrece, en el mejor de los casos, un fitness medio de 43.5.

A continuación se ofrecerá un gráfico representativo, que permitirá comparar los resultados obtenidos de un simple vistazo.



Ahora procederemos a estudiar el mejor caso de la programación genética, frente al mejor caso de la programación genética en árbol. Para esta comparación visualizaremos los mejores casos individuales, no promedios, que se corresponden con el experimento 2 repetición 4 para programación genética en árbol y el experimento 21 repetición 8 para la programación genética.

Programación genética

La configuración propia de este experimento se compone de los siguientes parámetros:

```
Experiment settings:
   prp_hypergp:off
   prp_generations:150
   prp_hgp_generations:110
```

Es decir: En esta ejecución, se han utilizado 150 generaciones de programación genética, porque al estar la programación genética en árbol desactivada, se ignora que le correspondiesen 110 generaciones.

La salida estándar de esta ejecución nos ofrece la siguiente información:

```
-----------------------
|-=    Generation 0 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             66.0000  |         0.0149  |          83  |           9  |
Generation Mean  |             79.6117  |         0.0124  |     68.7400  |      7.9900  |
```

```
  Worst of Gen.  |             80.0000  |         0.0123  |          79  |           7  |
-------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0000  |                 0.0000  |
Evaluation Time  |              5.4850  |              3291.0000  |
-------------------------------------------------------------------

 New Best Individual:

    Raw fitness: 66.0
    Adjusted fitness: 0.014925373134328358

       Tree  Nodes  Depth
    ---------------------
     Tree 0     83      9
    ---------------------

Tree 0:
    (Prog2     (Prog2     (ifLwEq     S11
                S4
                (Prog2     MF
                    (Prog2     (Prog2     (Prog2     R
                                L
                            )
                            (Prog2     MF
                                L
                            )
                        )
                        (ifLwEq     S2
                            S7
                            L
                            (ifLwEq     S1
                                S110
                                L
                                (ifLwEq     S5
                                    S6
                                    R
                                    L
                                )
                            )
                        )
                    )
                )
                MB
            )
            (Prog2     (Prog2     MB
                    MB
                )
                L
            )
        )
        (Prog2     (Prog2     (Prog2     (Prog2     (ifLwEq     S9
                            S7
                            MF
                            (Prog2     R
                                (Prog2     MB
```

```
                                MF
                            )
                        )
                    )
                    R
                )
                (Prog2     (ifLwEq     S9
                        S6
                        MB
                        (Prog2     MF
                            MB
                        )
                    )
                    (ifLwEq     S11
                        S110
                        R
                        MB
                    )
                )
            )
            (Prog2     MB
                (Prog2     (Prog2     (ifLwEq     S1
                            S6
                            R
                            (Prog2     (ifLwEq     SS
                                    S5
                                    L
                                    MF
                                )
                                MB
                            )
                        )
                        (Prog2     (Prog2     MB
                                L
                            )
                            MB
                        )
                    )
                    MB
                )
            )
        )
        L
    )
)
```

==========

En la primera generación siempre se obtiene un nuevo mejor individuo. La media de esta generación no es nada buena, su fitness es de 79.6117, casi el peor que se puede obtener. El mejor individuo, sin embargo tiene un fitness de 66.0000, considerablemente mejor. También se puede observar que el mejor individuo es el que tiene una mayor profundidad, lo cual tiene sentido de cara a la naturaleza del problema, en el que es importante ser capaz de encadenar una serie de movimientos encadenados.

```
----------------------
|-=    Generation 1 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             66.0000  |          0.0149  |           83  |            9  |
Generation Mean  |             78.8650  |          0.0125  |      68.3900  |       7.9600  |
  Worst of Gen.  |             80.0000  |          0.0123  |           83  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0333  |               20.0000  |
Evaluation Time  |              6.3200  |             3792.0000  |
================================================================================================
```

En esta primera generación no se ha conseguido mejorar el mejor individuo de la primera generación, sin embargo, sí que ha mejorado el fitness medio de la población, que ha pasado de 79.6117 a 78.8650.

```
----------------------
|-=    Generation 2 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             51.0000  |          0.0192  |           91  |            9  |
Generation Mean  |             77.2217  |          0.0128  |      67.1533  |       7.8700  |
  Worst of Gen.  |             80.0000  |          0.0123  |           91  |           10  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0367  |               22.0000  |
Evaluation Time  |              7.1467  |             4288.0000  |
-------------------------------------------------------------------

 New Best Individual:

    Raw fitness: 51.0
    Adjusted fitness: 0.019230769230769232

       Tree  Nodes  Depth
    ---------------------
     Tree 0     91      9
    ---------------------

Tree 0:
    (Prog2     (Prog2     (Prog2     (ifLwEq     S6
                    S7
                    (ifLwEq     S4
                        S5
                        (Prog2     (Prog2     MB
                                MB
                            )
                            (ifLwEq     S8
                                S110
                                MF
                                MB
                            )
                        )
                        L
                    )
```

```
            L
        )
        MB
    )
    MB
)
(Prog2    (ifLwEq    S6
        SS
        (ifLwEq    S7
            S6
            (ifLwEq    S5
                S7
                R
                MF
            )
            MB
        )
        (Prog2    (ifLwEq    S4
                S1
                MB
                MF
            )
            MB
        )
    )
    (Prog2    (ifLwEq    S2
            S0
            (Prog2    (ifLwEq    S9
                    SS
                    (Prog2    L
                        MB
                    )
                    MB
                )
                MB
            )
            (Prog2    L
                MF
            )
        )
        (ifLwEq    S4
            S2
            (ifLwEq    S6
                S7
                (ifLwEq    S3
                    S2
                    (Prog2    MB
                        L
                    )
                    MF
                )
                (ifLwEq    S1
                    S9
                    (Prog2    MB
                        (ifLwEq    S110
                            S0
                            L
                            (ifLwEq    S110
                                S5
                                L
                                MB
                            )
                        )
                    )
                    MB
                )
            )
            (ifLwEq    S110
```

```
                    S4
                    MF
                    MB
                )
            )
        )
    )
)
```

```
================================================================================
```

En esta segunda generación se ha vuelto a obtener un nuevo mejor individuo, pasando de un fitness de 66.0000 a uno de 51.0000. A su vez el fitness medio de la población sigue mejorando, pasando de 78.8650 a 77.2217. En estas primeras generaciones se produce una rápida mejora de los individuos, pero según se vaya mejorando, cada vez costará más trabajo obtener mejores individuos.

```
----------------------
|-=    Generation 3 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             51.0000  |         0.0192  |           91  |            9  |
Generation Mean  |             74.7050  |         0.0133  |      67.4467  |       7.8817  |
  Worst of Gen.  |             80.0000  |         0.0123  |           43  |            7  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------
  Breeding Time  |              0.0333  |               20.0000  |
Evaluation Time  |              8.2733  |             4964.0000  |
================================================================================================

. . .

----------------------
|-=   Generation 10 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             51.0000  |         0.0192  |           91  |            9  |
Generation Mean  |             65.7217  |         0.0150  |      87.7167  |       8.9967  |
  Worst of Gen.  |             70.0000  |         0.0141  |           67  |            8  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------
  Breeding Time  |              0.0533  |               32.0000  |
Evaluation Time  |             14.0367  |             8422.0000  |
================================================================================================


----------------------
|-=   Generation 11 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             51.0000  |         0.0192  |           91  |            9  |
Generation Mean  |             65.7450  |         0.0150  |      88.0800  |       8.9983  |
  Worst of Gen.  |             71.0000  |         0.0139  |           91  |            9  |
```

```
--------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0333  |               20.0000  |
Evaluation Time  |             14.1417  |             8485.0000  |
================================================================================
```

Según se han ido computando generaciones, el fitness medio de la población ha pasado de 74.7050 a 65.7450. De esta forma, cada vez la generación de nuevos individuos tendrá un mejor conjunto inicial.

Cabe destacar que entre la generación 10 y la 11 de ha producido un empeoramiento del fitness medio de la población. Esto se debe a que una de las claves de la programación genética es la variabilidad de la población, y según se vaya mejorando la población, se puede producir un empeoramiento del fitness medio, sin embargo los mejores individuos de la población se conservan.

```
-----------------------
|-=  Generation 150 =-|
================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
--------------------------------------------------------------------------------
   Best of Gen.  |             51.0000  |         0.0192  |          91  |           9  |
Generation Mean  |             66.0000  |         0.0149  |     90.6933  |      9.0000  |
  Worst of Gen.  |             66.0000  |         0.0149  |          91  |           9  |
--------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0333  |               20.0000  |
Evaluation Time  |             11.8833  |             7130.0000  |
================================================================================
```

Finalmente el sistema no fue capaz de generar un mejor individuo. Puesto que los individuos son de 91 nodos y la limitación por configuración es de 100, es posible que eso haya limitado la evolución. Otro aspecto a tener en cuenta es que las poblaciones estaban formadas únicamente por 600 individuos, mientras que en otros experimentos similares se han hecho con 4000 individuos.

Programación genética en árbol

La programación genética en árbol se basa en esquemas, es decir, maneja individuos arbóreos que realmente representan un subconjunto de soluciones en lugar de una única solución. Se considera que un individuo es mejor que otro cuando el conjunto de sus fitness es mejor que el conjunto de fitness de otros individuos.

Un problema de esta técnica sería la evaluación de las soluciones pertenecientes al subconjunto marcado por el individuo arbóreo. La evaluación de todas las soluciones supondría un coste computacional demasiado elevado, por lo que la evaluación de individuos arbóreos se realiza mediante un muestreo de los individuos que representa, con lo que se obtendrá una estimación estadística de su fitness.

```
-----------------------
|-=    Generation 0 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             75.0000  |         0.0132  |           57  |            9  |
Generation Mean  |             79.6740  |         0.0124  |      68.2267  |       7.9583  |
  Worst of Gen.  |             80.0000  |         0.0123  |           49  |            8  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0000  |                 0.0000  |
Evaluation Time  |             51.9900  |            31194.0000  |
-------------------------------------------------------------------

 New Best Individual:

    Raw fitness: 75.0
    Adjusted fitness: 0.013157894736842105

       Tree  Nodes  Depth
    ---------------------
     Tree 0     57      9
    ---------------------

Tree 0:
    (=     (=     =
            =
            =
            (=     =
                (=     =
                    (=     =
                        =
                    )
                )
            )
        )
        (=     =
            =
            (=     (=     =
                    =
                )
                (=     (=     (=     =
                            =
                            =
                            (=     =
                                (=     =
                                    =
                                    =
                                    (=     =
                                        =
                                    )
                                )
                            )
                        )
```

```
                    (=     =
                        =
                        (=     =
                            (=     (=     =
                                    =
                                    =
                                    =
                                )
                                =
                            )
                        )
                        (=     =
                            =
                            (=     =
                                =
                            )
                            (=     =
                                =
                            )
                        )
                    )
                )
                =
            )
        )
        =
    )
)
```

```
================================================================================================
```

En la primera generación, el mejor individuo arboreo de la población inicial es el mostrado anteriormente. En este se puede observar la forma que va tomando, creciendo más en profundidad que en anchura. El fitness que ha dado como resultado el muestreo de los individuos ha sido de 75.0000.

```
-----------------------
|-=    Generation 2 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             74.8000  |         0.0132  |          65  |           8  |
Generation Mean  |             79.0769  |         0.0125  |     66.0100  |      8.2533  |
  Worst of Gen.  |             80.0000  |         0.0123  |          83  |          10  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0983  |               59.0000  |
Evaluation Time  |             41.4767  |            24886.0000  |
-------------------------------------------------------------------

 New Best Individual:

    Raw fitness: 74.8
    Adjusted fitness: 0.013192612137203167

       Tree  Nodes  Depth
    ---------------------
     Tree 0     65      8
    ---------------------
```

```
Tree 0:
    (=     (=     (=     (=     =
                    (=     =
                        (=     =
                            =
                        )
                    )
                )
                =
            )
            (=     (=     =
                =
                =
                (=     =
                    =
                    =
                    =
                )
            )
            (=     =
                =
            )
        )
    )
    (=     (=     =
            (=     =
                (=     (=     =
                        =
                    )
                    (=     =
                        =
                        =
                        (=     =
                            (=     =
                                =
                                =
                                =
                            )
                        )
                    )
                )
            )
            (=     =
                =
                (=     =
                    (=     (=     =
                            (=     =
                                =
                            )
                            =
                        )
                    )
                )
                (=     =
                    =
                    (=     =
                        =
                    )
                    =
                )
            )
        )
    )
```

En la segunda generación podemos observar que tanto se ha producido una mejora en el fitness medio de la población, como que se ha producido un nuevo mejor individuo, aunque en este caso, la mejora ha sido considerablemente más significativa para la población general, que ha bajado casi un punto su fitness, que para el mejor individuo, que tan solo ha bajado un 0.2 su fitness.

```
       ...

----------------------
|-=    Generation 5 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             72.2000  |         0.0137  |          69  |           9  |
Generation Mean  |             78.5210  |         0.0126  |     68.7033  |      8.6733  |
  Worst of Gen.  |             80.0000  |         0.0123  |          93  |          11  |
------------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.1200  |               72.0000  |
Evaluation Time  |             46.3883  |            27833.0000  |
-----------------------------------------------------------------

 New Best Individual:

    Raw fitness: 72.2
    Adjusted fitness: 0.01366120218579235

       Tree  Nodes  Depth
    ---------------------
     Tree 0     69      9
    ---------------------

Tree 0:
    (=     (=     (=     (=     =
                    (=     =
                        (=     =
                            =
                        )
                    )
                )
                =
            )
            (=     (=     =
                    =
                    =
                    (=     =
                        =
                        =
                        =
                    )
                )
                (=     =
                    =
                )
            )
        )
        (=     (=     =
```

```
                (=     =
                    (=     (=     (=     =
                                =
                                =
                                (=     (=     =
                                            =
                                            =
                                            =
                                        )
                                        =
                                    )
                                )
                                =
                            )
                            (=     =
                                =
                                =
                                (=     =
                                    (=     =
                                        =
                                        =
                                        =
                                    )
                                )
                            )
                        )
                    )
                )
            )
            (=     =
                =
                (=     =
                    (=     =
                        (=     (=     =
                                (=     =
                                    =
                                )
                            )
                            =
                        )
                    )
                )
                =
            )
        )
    )
```

```
================================================================================
```

En estas primeras ejecuciones, se ha producido una sucesiva mejora del mejor individuo de la población, que ha ido cambiando en cada generación durante estas primeras 5 generaciones. Durante esta evolución, el mejor individuo ha ido probando distintas formas de árbol, finalmente adoptando la que mejor se adapta a la solución del problema.

```
-----------------------
|-=    Generation 6 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             72.2000  |         0.0137  |           69  |            9  |
Generation Mean  |             78.4038  |         0.0126  |      65.0600  |       8.8767  |
```

```
  Worst of Gen.  |             80.0000  |         0.0123  |          95  |           9  |
-------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0867  |               52.0000  |
Evaluation Time  |             46.8900  |            28134.0000  |
=================================================================================================

        ...


-----------------------
|-=    Generation 9 =-|
=================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
-------------------------------------------------------------------------------------------------
   Best of Gen.  |             69.6000  |         0.0142  |          93  |          10  |
Generation Mean  |             78.0990  |         0.0126  |     74.0400  |      9.5017  |
  Worst of Gen.  |             80.0000  |         0.0123  |          61  |           9  |
-------------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.1250  |               75.0000  |
Evaluation Time  |             53.6800  |            32208.0000  |
-----------------------------------------------------------------

 New Best Individual:

    Raw fitness: 69.6
    Adjusted fitness: 0.0141643059490085

       Tree  Nodes  Depth
    ---------------------
     Tree 0     93     10
    ---------------------

Tree 0:
    (=     (=     (=     (=     =
                    (=     =
                        (=     =
                            =
                        )
                    )
                )
                (=     (=     =
                        =
                        (=     =
                            (=     =
                                (=     (=     =
                                        =
                                        =
                                        (=     =
                                            =
                                        )
                                    )
                                    =
                                )
                            )
                        )
                        (=     =
                            =
                        )
                    )
```

```
                    =
                )
            )
            (=      (=      =
                    =
                    =
                    (=      =
                        =
                        =
                        =
                    )
                )
                (=      =
                    =
                )
            )
        )
        (=      (=      =
                (=      =
                    (=      (=      (=      =
                                =
                            )
                            =
                        )
                        (=      =
                            (=      =
                                (=      =
                                    =
                                )
                            )
                        )
                    )
                )
            )
            (=      =
                =
                (=      =
                    (=      =
                        (=      (=      =
                            =
                            =
                            (=      =
                                =
                            )
                        )
                        =
                    )
                )
                (=      =
                    (=      =
                        =
                        (=      =
                            (=      =
                                =
                            )
                            =
                        )
                    )
                    =
                )
            )
        )
    )
)
```

```
================================================================================
```

En este caso se ha podido observar un conjunto de generaciones en los que no se ha producido la mejora del mejor individuo, sin embargo, la población sí que ha ido evolucionando, pasando de un fitness de 78.4038 a uno de 78.0990. Al llegar a este punto, inevitablemente se ha producido una nueva mejora en el mejor individuo, que alcanza un fitness de 69.6000.

```
----------------------
|-=   Generation 10 =-|
================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
--------------------------------------------------------------------------------
   Best of Gen.  |             69.6000  |         0.0142  |          93  |          10  |
Generation Mean  |             77.9891  |         0.0127  |     74.7300  |      9.7217  |
  Worst of Gen.  |             80.0000  |         0.0123  |          27  |          11  |
--------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0983  |               59.0000  |
Evaluation Time  |             52.7383  |            31643.0000  |
================================================================================
```

Finalmente se completa la ejecución de la programación genética en árbol con un fitness de 69.6000 tras 10 generaciones.

```
----------------------
|-=    Generation 0 =-|
================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
--------------------------------------------------------------------------------
   Best of Gen.  |             31.0000  |         0.0313  |          93  |          10  |
Generation Mean  |             77.7683  |         0.0128  |     93.0000  |     10.0000  |
  Worst of Gen.  |             80.0000  |         0.0123  |          93  |          10  |
--------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0983  |               59.0000  |
Evaluation Time  |             12.2917  |             7375.0000  |
-----------------------------------------------------------------

 New Best Individual:

    Raw fitness: 31.0
    Adjusted fitness: 0.03125

       Tree  Nodes  Depth
    ---------------------
     Tree 0     93     10
    ---------------------

Tree 0:
    (Prog2     (Prog2     (Prog2     (Prog2     MB
                    (Prog2     L
                        (Prog2     L
```

```
                    MF
                )
            )
        )
        (Prog2    (ifLwEq    S3
                S7
                (Prog2    MB
                    (Prog2    MB
                        (Prog2    (ifLwEq    S1
                                S6
                                R
                                (Prog2    L
                                    L
                                )
                            )
                            MB
                        )
                    )
                )
                (Prog2    MF
                    MF
                )
            )
            MF
        )
    )
    (Prog2    (ifLwEq    S4
            S2
            L
            (ifLwEq    S6
                S8
                MB
                R
            )
        )
        (Prog2    R
            MB
        )
    )
)
(Prog2    (Prog2    MB
        (Prog2    MB
            (Prog2    (Prog2    (Prog2    L
                        MB
                    )
                    MB
                )
                (Prog2    R
                    (Prog2    R
                        (Prog2    MB
                            MB
                        )
                    )
                )
            )
        )
    )
    (ifLwEq    S2
        S7
        (Prog2    MB
            (Prog2    R
                (Prog2    (ifLwEq    S4
                        S7
                        MB
                        (Prog2    MB
                            R
                        )
                    )
```

```
                        L
                    )
                )
            )
            (Prog2     MB
                (ifLwEq     S3
                    S4
                    (Prog2     MF
                        (ifLwEq     S0
                            S0
                            (Prog2     R
                                L
                            )
                            MB
                        )
                    )
                    R
                )
            )
        )
    )
    )
```

```
==========================================================================================


----------------------
|-=    Generation 1 =-|
==========================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------
   Best of Gen.  |             31.0000  |         0.0313  |          93  |          10  |
Generation Mean  |             74.6583  |         0.0133  |     93.0000  |     10.0000  |
  Worst of Gen.  |             80.0000  |         0.0123  |          93  |          10  |
------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0500  |               30.0000  |
Evaluation Time  |             12.4283  |             7457.0000  |
==========================================================================================


. . .

----------------------
|-=  Generation 100 =-|
==========================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------
   Best of Gen.  |             31.0000  |         0.0313  |          93  |          10  |
Generation Mean  |             57.5000  |         0.0171  |     93.0000  |     10.0000  |
  Worst of Gen.  |             59.0000  |         0.0167  |          93  |          10  |
------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0500  |               30.0000  |
Evaluation Time  |             11.9700  |             7182.0000  |
==========================================================================================
```

Como se puede observar, la programación genética, pese a que llega a mejorar el fitness de la población, no consigue obtener un mejor individuo que el obtenido por la

programación genética en árbol. Esto no se puede achacar a que la configuración del problema sea inválida, porque en otras repeticiones la programación genética ha funcionado correctamente, sino que debe considerarse que la complejidad del problema es muy elevada y que las condiciones en las que se ha buscado una solución eran insuficientes.

De esta ejecución queremos destacar como la estructura del árbol obtenida durante la ejecución de la programación genética en árbol ha dado la forma al mejor individuo que se generó durante la generación 0 de la fase de programación genética.

Programación genética en árbol	**Programación genética**
Tree 0: (= (= (= (= = (= = (= = =))) (= (= = = (= = (= = (= (= = = = (= = =)) =))) =)) (= (= = = = (= = = = =)) (= = =))) (= (= = (= = (= (= (= = =) =	Tree 0: (Prog2 (Prog2 (Prog2 (Prog2 MB (Prog2 L (Prog2 L MF))) (Prog2 (ifLwEq S3 S7 (Prog2 MB (Prog2 MB (Prog2 (ifLwEq S1 S6 R (Prog2 L L)) MB))) (Prog2 MF MF)) MF)) (Prog2 (ifLwEq S4 S2 L (ifLwEq S6 S8 MB R)) (Prog2 R MB))) (Prog2 (Prog2 MB (Prog2 MB (Prog2 (Prog2 (Prog2 L MB) MB

```
          )
         (=   =
           (=   =
             (=   =
               =
              )
             )
            )
           )
          )
         )
        (=   =
          =
         (=   =
           (=   =
             (=  (=   =
                  =
                  =
                 (=   =
                    =
                  )
                 )
                 =
            )
           )
          )
         (=   =
           (=   =
             =
            (=   =
              (=   =
                 =
                (=   =
                   =
                 )
                 =
              )
             )
             =
           )
          )
         )
        )
       )
```

```
            )
            (Prog2   R
              (Prog2   R
                (Prog2   MB
                  MB
                )
              )
            )
          )
        )
      )
      (ifLwEq   S2
        S7
        (Prog2   MB
          (Prog2   R
            (Prog2   (ifLwEq   S4
                S7
                MB
                (Prog2   MB
                  R
                )
              )
              L
            )
          )
        )
        (Prog2   MB
          (ifLwEq   S3
            S4
            (Prog2   MF
              (ifLwEq   S0
                S0
                (Prog2   R
                  L
                )
                MB
              )
              R
            )
          )
        )
      )
    )
  )
```

Como se puede observar en la tabla superior, la estructura obtenida durante la programación genética es la que ha ofrecido una mejor solución al problema. La forma del árbol es exactamente la misma, tan solo se han reemplazado los operadores comodín (=) de la programación genética en árbol por operadores de la programación genética. A pesar de que en esta repetición, la programación genética no ha conseguido aportar nada, en otras ejecuciones si que lo ha hecho, partiendo de una base de programación genética en árbol, como es el caso del experimento 6 repetición 3, entre otros:

```
-----------------------
|-=   Generation 10 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |            71.0000  |         0.0139  |           81  |           11  |
Generation Mean  |            78.0634  |         0.0127  |      66.8433  |       9.9767  |
```

```
  Worst of Gen.  |             80.0000  |         0.0123  |          55  |          12  |
-------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0983  |               59.0000  |
Evaluation Time  |             55.7383  |            33443.0000  |
-----------------------------------------------------------------

 New Best Individual:

    Raw fitness: 71.0
    Adjusted fitness: 0.013888888888888888

       Tree  Nodes  Depth
    ---------------------
     Tree 0     81     11
    ---------------------

Tree 0:
    (=     =
        (=     =
            (=     (=     =
                    (=     (=     =
                            (=     (=     (=     (=     =
                                                  =
                                                  =
                                                  =
                                            )
                                            =
                                        )
                                        =
                                    )
                                    =
                                )
                            )
                            (=     =
                                =
                                =
                                =
                            )
                        )
                    )
                    (=     (=     (=     =
                                =
                            (=     (=     =
                                =
                            )
                            =
                        )
                    )
                    (=     =
                        =
                    )
                )
                (=     =
                    (=     (=     (=     =
                                =
                            )
                            (=     =
                                =
                                =
                                (=     =
                                    =
                                    (=     (=     =
                                            =
                                            =
                                            =
                                        )
                                        (=     =
                                            =
                                        )
                                    )
                                    =
```

```
                                )
                            )
                        )
                        (=     (=     (=     (=     =
                                 =
                                    )
                                    =
                                )
                                (=     =
                                    =
                                    =
                                    (=     (=     =
                                             =
                                             =
                                             =
                                         )
                                         =
                                    )
                                )
                            )
                            =
                        )
                    )
                )
            )
        )
    )
```

```
================================================================================================

-----------------------
|-=    Generation 0 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             39.0000  |          0.0250  |           81  |           11  |
Generation Mean  |             78.1433  |          0.0127  |      81.0000  |      11.0000  |
  Worst of Gen.  |             80.0000  |          0.0123  |           81  |           11  |
------------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0983  |               59.0000  |
Evaluation Time  |             14.7583  |             8855.0000  |
-----------------------------------------------------------------

 New Best Individual:

    Raw fitness: 39.0
    Adjusted fitness: 0.025

       Tree  Nodes  Depth
    ---------------------
     Tree 0     81     11
    ---------------------

Tree 0:
    (Prog2     MB
        (Prog2     MB
            (Prog2     (Prog2     L
                    (Prog2     (Prog2     L
                            (Prog2     (Prog2     (Prog2     (ifLwEq     S5
                                                      S1
                                                      L
                                                      L
                                                  )
                                                  R
                                              )
                                              R
```

```
                        )
                        L
                    )
                )
                (ifLwEq     S6
                    S5
                    L
                    R
                )
            )
        )
        (Prog2     (Prog2     (Prog2     R
                    (Prog2     (Prog2     MB
                            MB
                        )
                        R
                    )
                )
                (Prog2     MB
                    R
                )
            )
            (Prog2     MB
                (Prog2     (Prog2     (Prog2     L
                            MB
                        )
                        (ifLwEq     S1
                            SS
                            R
                            (ifLwEq     S9
                                S1
                                (Prog2     (ifLwEq     S5
                                        S9
                                        R
                                        L
                                    )
                                    (Prog2     MB
                                        MB
                                    )
                                )
                                MF
                            )
                        )
                    )
                    (Prog2     (Prog2     (Prog2     (Prog2     MB
                                    R
                                )
                                R
                            )
                            (ifLwEq     S3
                                S7
                                L
                                (Prog2     (ifLwEq     S9
                                        S110
                                        R
                                        MF
                                    )
                                    MF
                                )
                            )
                        )
                        MB
                    )
                )
            )
        )
    )
)
```

```
    )

================================================================================================


----------------------
|-=    Generation 1 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             36.0000  |         0.0270  |          81  |          11  |
Generation Mean  |             75.1583  |         0.0132  |     81.0000  |     11.0000  |
  Worst of Gen.  |             80.0000  |         0.0123  |          81  |          11  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------------
  Breeding Time  |              0.0467  |               28.0000  |
Evaluation Time  |             13.6450  |             8187.0000  |
-------------------------------------------------------------------

 New Best Individual:

    Raw fitness: 36.0
    Adjusted fitness: 0.02702702702702703

       Tree  Nodes  Depth
    ---------------------
     Tree 0     81     11
    ---------------------

Tree 0:
    (Prog2     MB
        (Prog2     MB
            (Prog2     (Prog2     L
                    (Prog2     (Prog2     L
                            (Prog2     (Prog2     (Prog2     (ifLwEq     S5
                                                        S1
                                                        L
                                                        L
                                                    )
                                                    R
                                                )
                                                R
                                            )
                                            L
                                    )
                                )
                                (ifLwEq     S6
                                    S5
                                    L
                                    R
                                )
                            )
                        )
                    (Prog2     (Prog2     (Prog2     R
                                (Prog2     (Prog2     MB
                                                MB
                                            )
                                            R
                                    )
                                )
                                (Prog2     MB
                                    R
                                )
                            )
                        (Prog2     MB
```

```
(Prog2     (Prog2     (Prog2     L
                MB
            )
            (ifLwEq     S1
                SS
                R
                (ifLwEq     S9
                    S1
                    (Prog2     (ifLwEq     S5
                            S9
                            R
                            L
                        )
                        (Prog2     MB
                            MB
                        )
                    )
                    MF
                )
            )
        )
        (Prog2     (Prog2     (Prog2     (Prog2     MB
                        R
                    )
                    R
                )
                (ifLwEq     S3
                    S7
                    L
                    (Prog2     (ifLwEq     S9
                            S110
                            R
                            MF
                        )
                        MF
                    )
                )
            )
            MB
        )
    )
)
)
)
)
)
```

==========

```
-----------------------
|-=    Generation 2 =-|
==========================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------
   Best of Gen.  |             34.0000  |          0.0286  |           81  |           11  |
Generation Mean  |             70.1683  |          0.0143  |      81.0000  |      11.0000  |
  Worst of Gen.  |             80.0000  |          0.0123  |           81  |           11  |
------------------------------------------------------------------------------------------

-----------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-----------------------------------------------------------------
  Breeding Time  |              0.0483  |               29.0000  |
Evaluation Time  |             15.2750  |             9165.0000  |
-----------------------------------------------------------------

 New Best Individual:
```

```
    Raw fitness: 34.0
    Adjusted fitness: 0.02857142857142857

       Tree  Nodes  Depth
    --------------------
     Tree 0     81     11
    --------------------

Tree 0:
    (Prog2     MB
        (Prog2     MB
            (Prog2     (Prog2     L
                    (Prog2     (Prog2     L
                            (Prog2     (Prog2     (Prog2     (ifLwEq     S5
                                                        S1
                                                        L
                                                        L
                                                    )
                                                    R
                                                )
                                                R
                                            )
                                            L
                                    )
                                )
                                (ifLwEq     S6
                                    S5
                                    L
                                    R
                                )
                            )
                        )
                    )
                    (Prog2     (Prog2     (Prog2     R
                                (Prog2     (Prog2     MB
                                                MB
                                            )
                                            R
                                    )
                                )
                                (Prog2     MB
                                    R
                                )
                            )
                        )
                        (Prog2     MB
                            (Prog2     (Prog2     (Prog2     L
                                                MB
                                        )
                                        (ifLwEq     S1
                                            SS
                                            R
                                            (ifLwEq     S9
                                                S1
                                                (Prog2     (ifLwEq     S5
                                                                S9
                                                                R
                                                                L
                                                            )
                                                            (Prog2     MB
                                                                MB
                                                            )
                                                        )
                                                        MF
                                                )
                                            )
                                        )
                                        (Prog2     (Prog2     (Prog2     (Prog2     MB
                                                                    R
```

```
                                    )
                                    R
                                )
                                (ifLwEq     S3
                                    S7
                                    L
                                    (Prog2     (ifLwEq     S9
                                                S110
                                                R
                                                MF
                                            )
                                            MF
                                    )
                                )
                            )
                            MB
                        )
                    )
                )
            )
        )
    )
==========================================================================================
```

Como se ha podido observar en esta ejecución, el motor de programación genética ha sido capaz de continuar evolucionando hacia soluciones mejores, a pesar del gran avance que proporciono la programación genética en árbol.

La diferencia de rendimiento entre las dos técnicas de programación genética sugieren que el problema es demasiado complejo para poder evolucionar en las condiciones proporcionadas a la programación genética. Un detalle que también hay que tener en cuenta es el tiempo empleado por las distintas generaciones; mientras que la programación genética consume del orden de 9 segundos por generación, la programación genética en árbol requiere un tiempo del orden de 32 segundos por generación. Esta es una diferencia considerablemente alta, permitiendo aumentar el número de individuos de la población, en programación genética, para que las comparaciones sean más realistas.

Una vez hemos evaluado la comparación entre las ejecuciones de programación genética e arbórea, procederemos a realizar las mismas comparaciones de resultados que hicimos con el experimento anterior.

En primer lugar, veremos los resultados obtenidos en función del distinto número de generaciones asignadas a la parte de programación genética en árbol. Las tablas de datos obtenidas son las siguientes:

Ejecución con 10 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_0	55,900	49,000
exp_2	51,500	31,000
exp_4	55,700	49,000
exp_6	52,400	34,000

Ejecución con 60 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_8	46,700	41,000
exp_10	44,100	39,000
exp_12	43,500	37,000
exp_14	46,000	43,000

Ejecución con 110 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_16	44,700	40,000
exp_18	44,700	37,000
exp_20	46,500	42,000
exp_22	46,400	36,000

Ejecución con 160 Generaciones arbóreas

Experimento	Average Fitness	Best Fitness
exp_24	44,900	39,000
exp_26	47,100	40,000
exp_28	45,100	36,000
exp_30	43,500	40,000

Tabla 10 - Resultados del experimento 2 por número de generaciones arbóreas

Con las tablas anteriores se ha obtenido el siguiente gráfico comparativo:

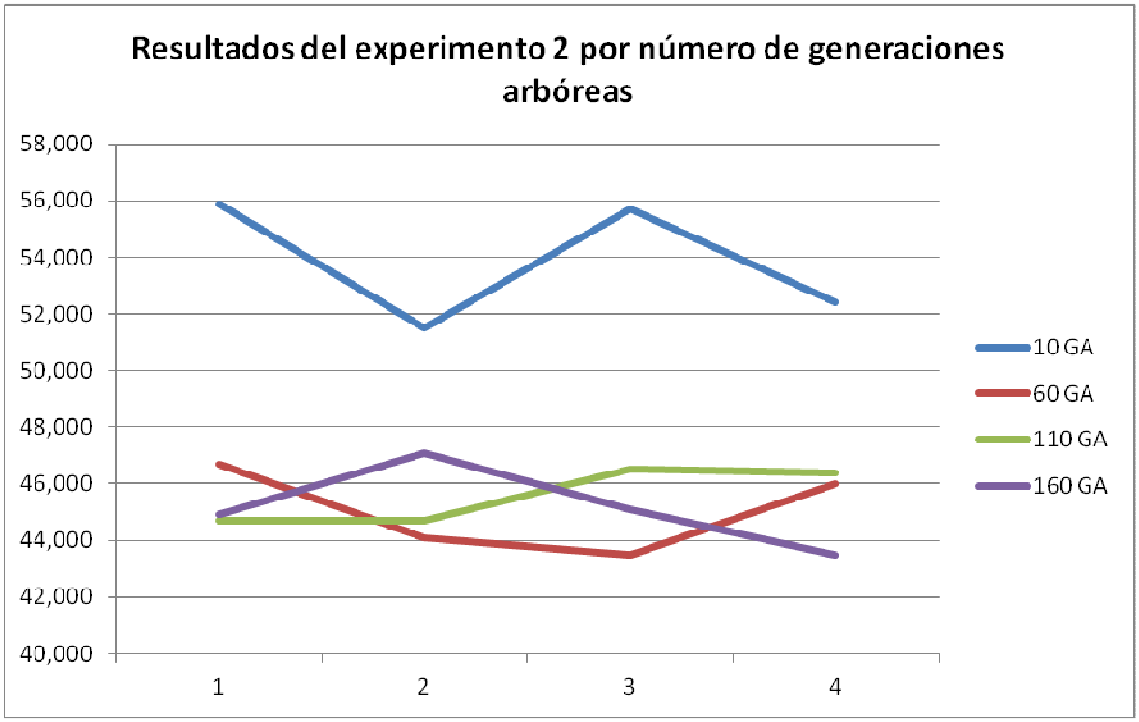


Tabla 11 - Gráfica de los resultados del experimento 2 por número de generaciones arbóreas

Como se puede observar, de nuevo se ha obtenido que 10 generaciones arbóreas son un número insuficiente, mientras que por el contrario se puede observar como ya no se produce una mejora al pasar de 60 a 160 generaciones arbóreas.

A continuación realizaremos la comparativa de la evolución según el número de generaciones de programación genética tradicional:

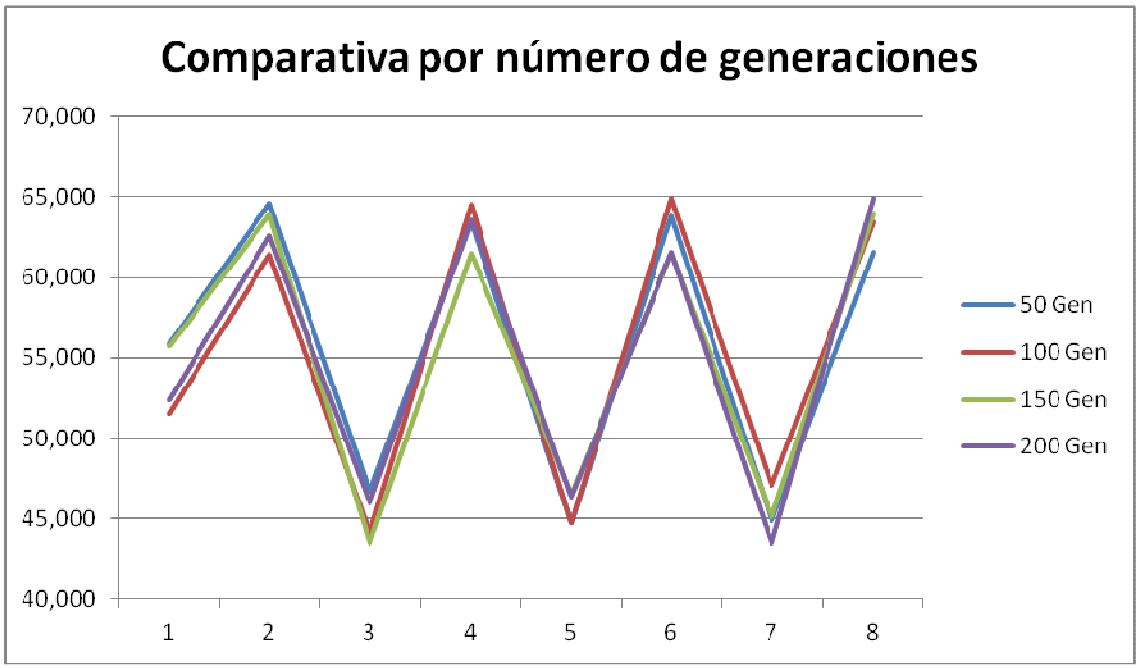


Tabla 12 - Gráfica de los resultados del experimento 2 por número de generaciones

Como se puede observar vuelve a darse el caso de que, como vimos antes, las ejecuciones con programación genética han obtenido mejores resultados independientemente de las generaciones de programación genética. Por ello, esta vez realizare dividiré la grafica en resultados con programación genética y con programación genética en árbol.

Los resultados con programación genética en árbol desactivada son:

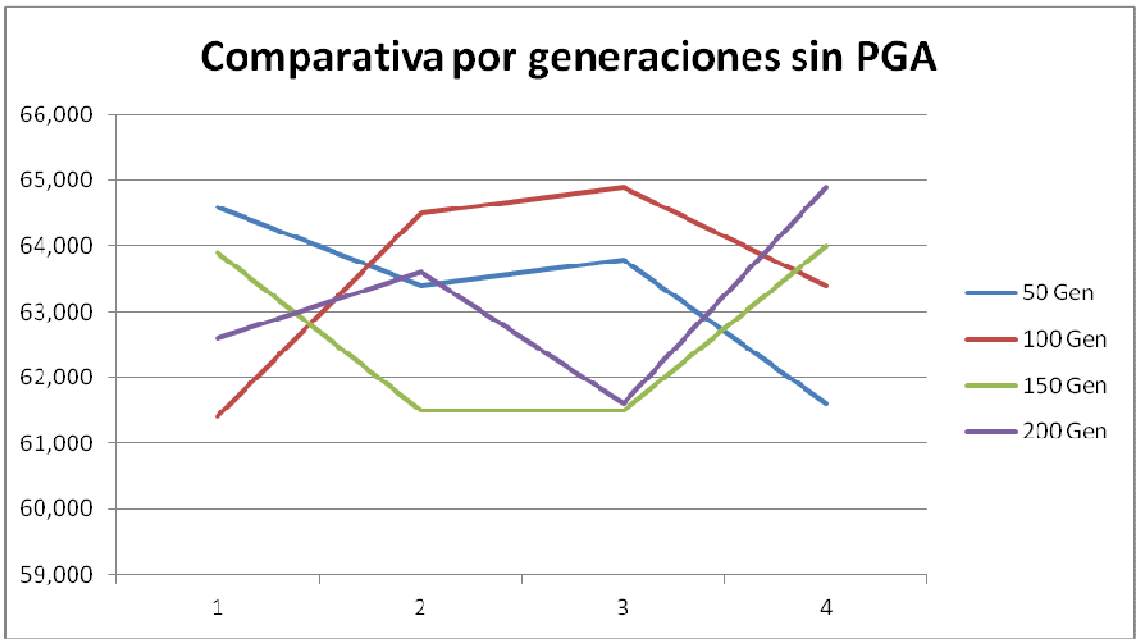


Tabla 13 - Gráfica de los resultados del experimento 2 por número de generaciones sin PGA

En la gráfica se representan, para cada número de generaciones, los cuatro valores de los cuatro experimentos que se han realizado para dicho número de generaciones. Recordemos que cada experimento, a su vez, ha constado de 10 repeticiones, por lo que al no haber un número que, de manera constante, de resultados mejores que los demás, podremos deducir que el número de generaciones no es un factor determinante a la hora de obtener buenos resultados.

Como se puede observar, las líneas que representan los distintos números de generaciones se entrecruzan, sin que ninguna de ellas ofrezca un resultado mejor que las otras. Por lo que se observó en la grafica general, algo similar se podá observar en la gráfica con programación genética en árbol activada.

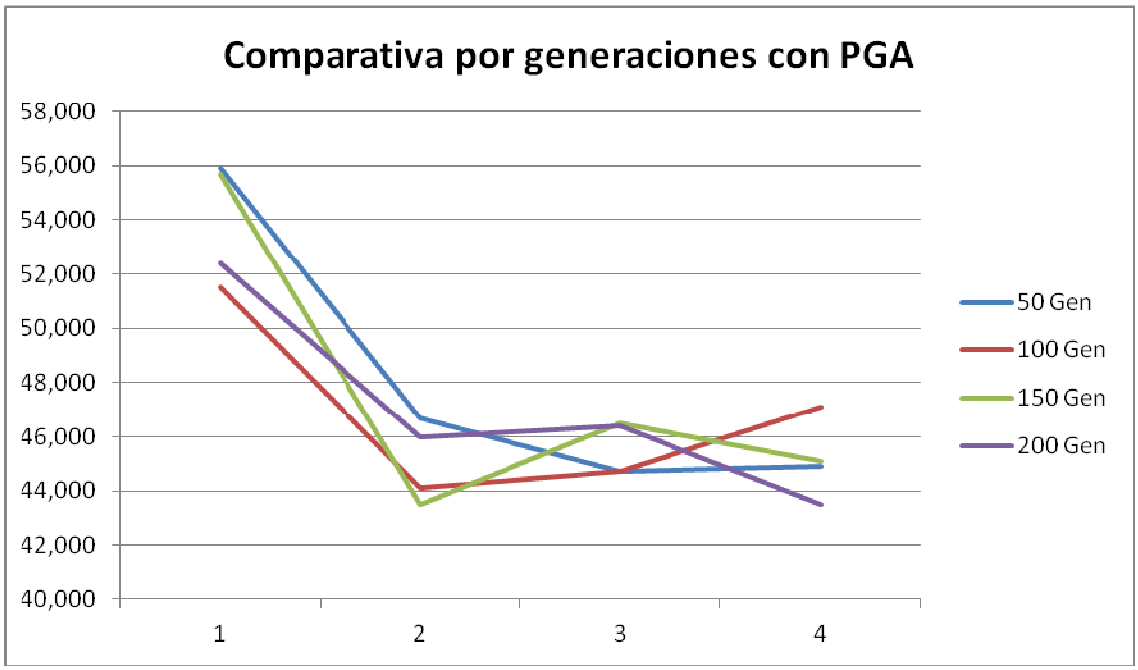
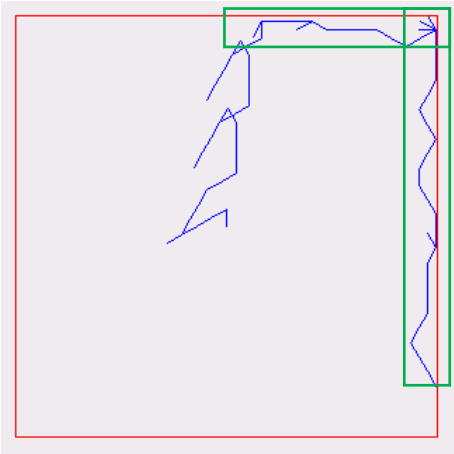


Tabla 14 - Gráfica de los resultados del experimento 2 por generaciones arbóreas

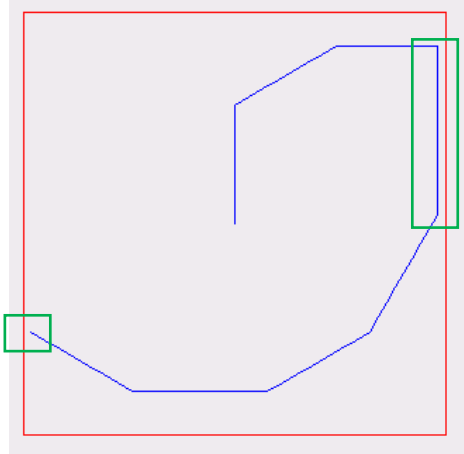
Lo primero que llama la atención de esta gráfica es el peor funcionamiento del primer caso. Esto se debe a que la columna 1 se corresponde con la ejecución con 10 generaciones de programación genética en árbol, que ofrece peores resultados. Obviando este hecho, el resto de resultados coinciden con lo observado anteriormente y no ofrecen una solución mejor.

3.3.2. Visualización de los resultados



E2 Mejor Mejor Robot con HGP.avi

Ilustración 33 - Mejor individuo del experimento 2 con PGA

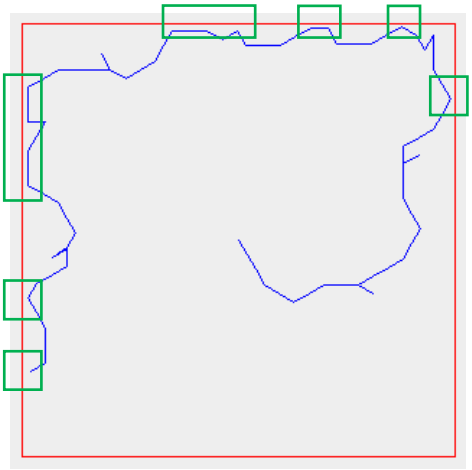


E2 Mejor Mejor Robot sin HGP.avi

Ilustración 34 - Mejor individuo del experimento 2 sin PGA

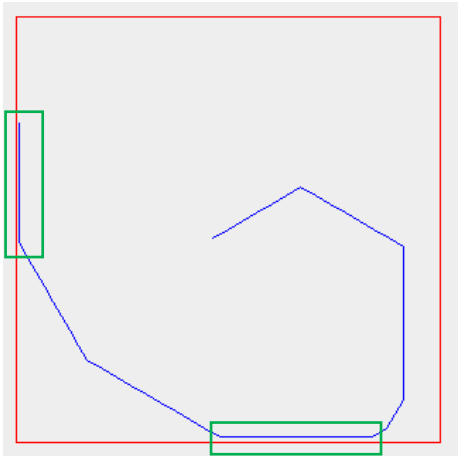
Como se puede observar, particularmente en el primer caso, se han conseguido unos resultados en los que se puede observar cómo el robot se dirige a las paredes de la habitación y trata de seguirlas, en lugar de moverse aleatoriamente por la habitación. De hecho en el primer caso se puede observar cómo el robot consigue hacer el giro en la esquina de la habitación de forma correcta y avanzar siguiendo la siguiente pared.

Finalmente, como en el experimento anterior, a continuación mostraremos los peores resultados de entre todas las baterías.



E2 Peor Mejor Robot con HGP.avi

Ilustración 35 - Peor individuo de las baterías de prueba del experimento 2 con PGA

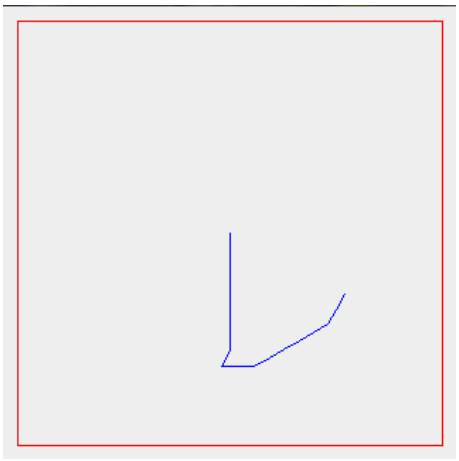


E2 Peor Mejor Robot sin HGP.avi

Ilustración 36 - Peor individuo de las baterías de prueba del experimento 2 sin PGA

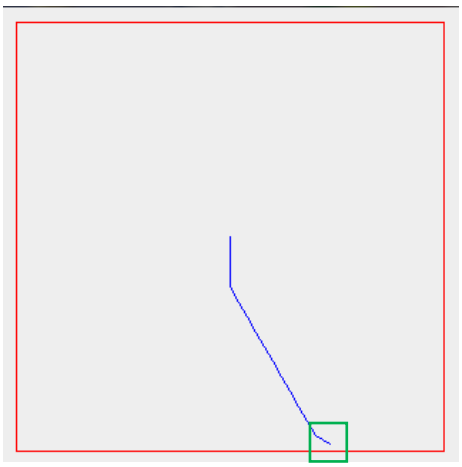
En los resultados anteriores se puede observar perfectamente la intención del seguimiento de las paredes, especialmente el robot que tiene activada la programación genética en árbol. En él se puede observar que aunque no consigue mantenerse suficientemente pegado a la pared, si que las va recorriendo, con mayor o menor grado de éxito.

Por último se mostrarán los peores resultados obtenidos de entre todas las repeticiones.



E2 Peor Peor Robot sin HGP.avi

Ilustración 37 - Peor individuo de las repeticiones del experimento 2 con sin PGA



E2 Peor Peor Robot con HGP.avi

Ilustración 38 - Peor individuo de las repeticiones del experimento 2 con PGA

En estos ejemplos se puede observar la complejidad real del problema, puesto que en el primer caso ni siquiera es capaz de llegar a la pared, mientras que en el segundo, aunque llega, no es capaz de comenzar a recorrerla siquiera.

3.3.3. Conclusiones

En este experimento, tampoco se ha conseguido que la programación genética funcionase correctamente.

Puesto que ya son dos los experimentos que la programación genética no ha sido capaz de evolucionar sensiblemente, es posible que las limitaciones de configuración sean la causa. Sin embargo, tampoco se puede decir que la programación genética se ha comportado de un modo ineficaz, se puede observar como la media de las poblaciones ha ido mejorando con el paso de las generaciones, pero aún así, apenas ha conseguido obtener nuevos mejores individuos.

La programación genética en árbol, por el contrario, si que ha conseguido evoluciones muy rápidas. Esto sería fácilmente justificable diciendo que cada individuo arbóreo de la población, en realidad representa a un subconjunto de individuos. De esta forma, aunque la población fuese únicamente de 600 individuos arbóreos, su equivalente en individuos sería mucho más alto.

En este experimento se ha vuelto a observar el claro predominio de la programación genética en árbol. Esta es la que ha obtenido los resultados más importantes, aunque, tanto en el experimento anterior como el actual hay que tener en cuenta el coste computacional de ambos sistemas.

Por ejemplo, comparando las ejecuciones de los 2 primeros experimentos, se obtienen los siguientes datos en milisegundos:

Experimento 1	
PGA	GP
804	489
894	563
1001	454
790	489
964	537
1038	504
1050	622
1216	629
1145	546
964	483
9866	5316
185,59067	

Es decir, los tiempos que ha tardado en ejecutar las repeticiones en total cuando ejecutaba programación genética en árbol ha sido de 9866 ms, mientras que cuando ejecutaba solo programación genética ha sido de 5315. Esto nos ofrece que las 10 generaciones de programación genética en árbol que ha ejecutado en cada repetición han supuesto una penalización en tiempo del 185,59%.

Ahora veremos el caso extremo, de diferencia computacional. En los experimentos 30 y 31 se ejecutan 200 generaciones de programación genética y 160 de programación genética en árbol. En este caso la tabla anterior queda:

Experimento 1	
PGA	GP
7058	1671
7004	1256
6885	1573
7292	1729
6986	1406
8831	1561
8215	1829
8371	1045
10000	1661
9611	1253
80253	14984
535,591297	

Como se puede observar, para una ejecución con 160 generaciones arbóreas se llega a una penalización de rendimiento del 535,59%. De esta forma, se puede concluir que aunque la programación genética en árbol ofrece mejores resultados, hay que tener en cuenta el coste computacional. Esto es especialmente sensible cuando se observa que incrementar el número de generaciones arbóreas no mejora el rendimiento, como se ha observado para 60, 110 y 160 generaciones arbóreas.

Respecto a los resultados gráficos, se puede observar que el primer resultado es significativamente mejor que el segundo resultado. Consigue ajustar su recorrido al contorno de la pared, únicamente sufriendo un poco al principio hasta llegar a la pared.

El segundo, en cambio emplea buena parte de su recorrido en cruzar la habitación sin acercarse a las paredes.

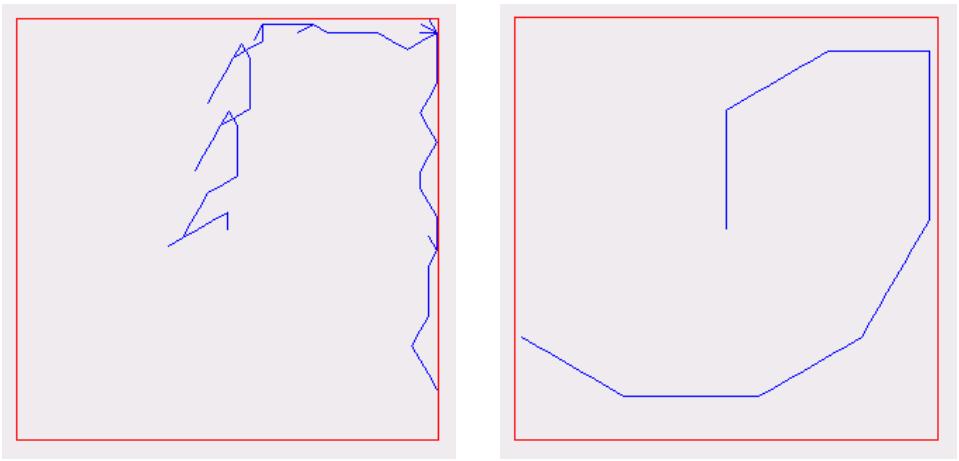


Ilustración 39 - Mejores resultados del experimento 2

En términos absolutos representa una solución peor que el primer resultado, puesto que realmente recorre menos puntos de la pared que el primero, aunque permite, mirando un poco más allá, ofrecer un resultado mucho mejor ejecutando el proceso el doble de veces.

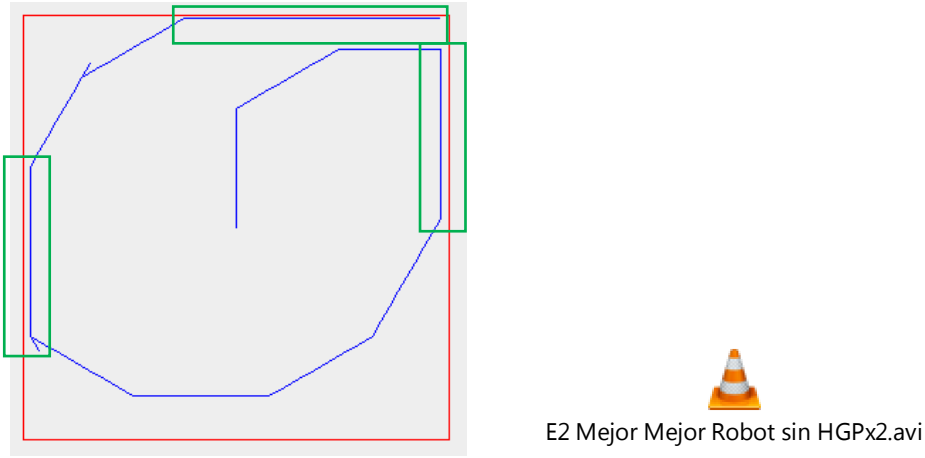


Ilustración 40 - Mejor individuo sin PGA del experimento 2 ejecutado el doble de veces

Como se puede observar el comportamiento es sensiblemente mejor únicamente aumentando el recorrido que puede realizar el robot.

No obstante, aunque esas sean las conclusiones obvias de este ejemplo se pueden extraer muestras de una mayor "inteligencia" de lo que se podría esperar. De hecho,

observaremos como los robots son inteligentes en cierto modo, no simplemente secuencias de acciones. Veamos como comienza la ejecución la primera vez y como comienza la segunda.

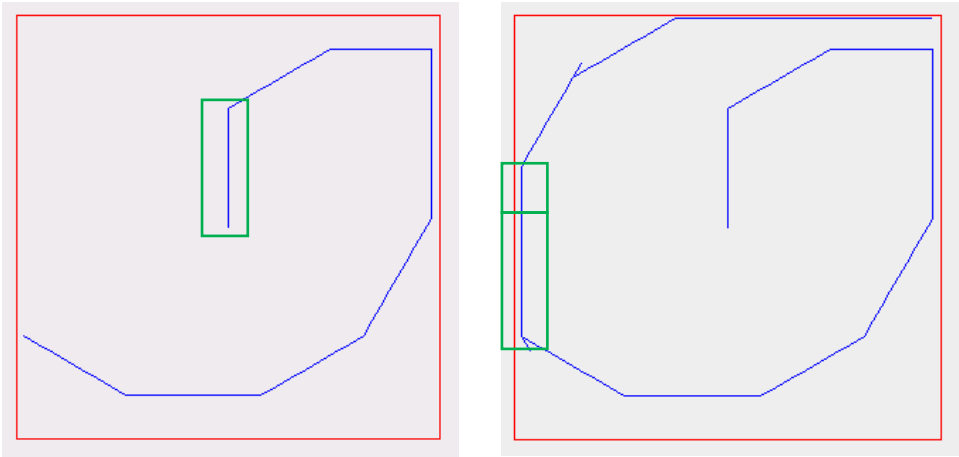


Ilustración 41 - Ejemplo 1 de adaptaciones de los individuos a su entorno

El comienzo de la ejecución es aparentemente idéntico. Prácticamente el robot avanza hacia arriba, como está marcado en verde. Sin embargo, la segunda ejecución el robot se da cuenta de que está avanzando pegado a una pared utilizando los sensores de que dispone y prosigue avanzando en esa dirección en vez de girar como ya había hecho en la primera ejecución.

Tenemos otro ejemplo de esto al final de la ejecución.

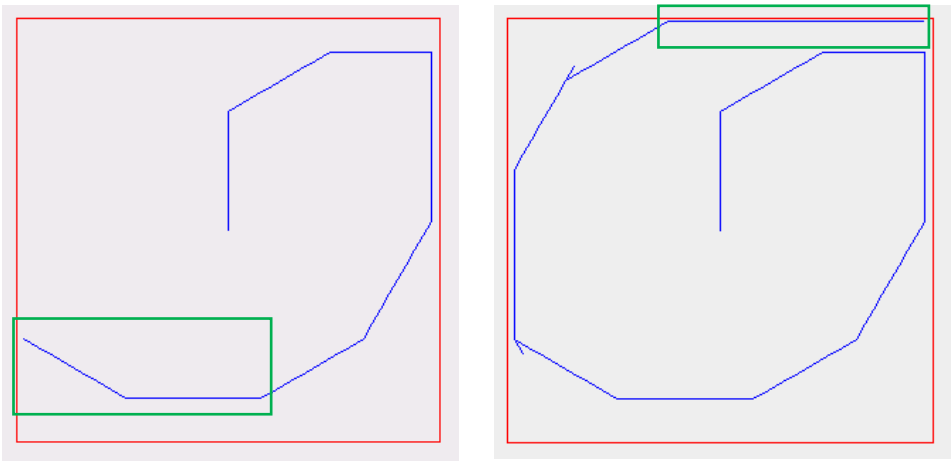


Ilustración 42 - Ejemplo 2 de adaptaciones de los individuos al entorno

En el primer caso, el robot avanza lejos de la pared, por lo que gira tratando de acercarse a una de ellas, sin embargo en el segundo caso el robot ya está recorriendo una pared, por lo que continua avanzando para mejorar sus puntuaciones.

En ambos casos es el mismo robot, uno que es capaz de adaptarse a la situación en la que se encuentra y ver qué es lo que más le conviene: si está pegado a una pared, avanzar recto para seguir marcando checkpoints, sino, girar para ver si la encuentra.

3.4. Experimento 3

Hasta ahora hemos podido comprobar el buen funcionamiento de la programación genética en árbol, que ha proporcionado algunos resultados bastante interesantes.

También hemos podido comprobar que mientras que la programación genética en árbol ha conseguido dar buenos resultados en un número determinado de generaciones arbóreas y, posteriormente, generaciones, la programación genética tradicional no ha sido capaz.

La programación genética ya ha reportado éxitos en campos como el que estamos tratando, pueden observarse los resultados de Koza en el Anexo 1. El hecho de que nosotros no hayamos obtenido unos resultados buenos puede tener varias causas; la primera de ellas la complejidad del problema, que es bastante probable que requiera un mayor número de individuos.

Para comprobar si el problema reside en la programación genética tradicional, realmente es ineficaz según la implementación de este problema, se ha realizado este experimento. En él se comprobará si la programación genética por si sola es capaz de generar un individuo aproximadamente tan bueno como el de la programación genética en árbol.

A continuación se describirá la configuración utilizada en ProGen para este experimento:

```
#Output: detailed or summarized
output_mode: detailed
prp_hypergp: on
prp_hgp_generations: 100
prp_hgp_convergence_limit: 95%
prp_hgp_sample_size: 5
prp_hgp_reevaluation_sample_size = 1

#----POPULATION----
#You can load the initial population from an XML file if you dont want it to be generated randomly
prp_load_population: population.xml
prp_population_size: 4000
prp_random_seed: 1234567890
#Full, grow, half and half
prp_initialization_mode: grow
#Valid depth range for the initial population trees
prp_depth_interval: 4, 10
#Maximun number of attempts to generate a valid tree (both for the initial population and during the evolution)
prp_max_attempts: 1000

#----INDIVIDUALS----
#valid trees during the evolution
prp_max_nodes: 200
prp_max_depth: 50
```

```
#----FUNCTIONS and ADF'S----

#if you want to use ADF'S name them like ADF0, ADF1 and so. To refer its branches use ARG0, ARG1...
prp_num_function_sets: 1
prp_function_set_0: Prog2Fc, IfLwEqualsThanFc, S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, SS, MoveForeward,
MoveBackward, TurnLeft, TurnRight
prp_return_type_fs_0: void
prp_function_set_1: AntMove, AntIfFoodAheadFc, Prog2Fc, ARG0
prp_return_type_fs_1: void
prp_number_of_trees: 1
prp_tree0_function_set_number: 0
prp_number_of_adfs: 0
prp_ADF0_function_set_number: 1
prp_ADF0_interface: void$$void

#----EVOLUTION----
prp_generations: 200
#Evolution will stop when this value (or any other inside the allowed error interval) is reached
#prp_stop_fitness: 0
#Upper and lower bounds from the stop_fitness value. Reaching a value in this interval will also stop the evolution
#prp_error_interval: 5, 0
#checkpoints. Valid values are either a list of integers or/and the word "every" followed by an integer.
prp_checkpoints: 10, 47, 82, every 100
#exact number or percentage value
prp_elitism: 2%

#Genetic operators to be applied
prp_GP_operator_number: 3
prp_GP_op1_name: OnePointCrossover(internal=0.8)
prp_GP_op1_probability: 0.55
prp_GP_op1_selection: Tournament(size=4)

prp_GP_op2_name: Reproduction(internal=1.0)
prp_GP_op2_probability: 0.05
prp_GP_op2_selection: Roulette

prp_GP_op3_name: PointMutation(internal=1)
prp_GP_op3_probability: 0.40
prp_GP_op3_selection: Tournament(size=4)

prp_HGP_operator_number: 3
prp_HGP_op1_name: Crossover(internal=0.8)
prp_HGP_op1_probability: 0.80
prp_HGP_op1_selection: Tournament(size=10)

prp_HGP_op2_name: Reproduction(internal=1.0)
prp_HGP_op2_probability: 0.10
prp_HGP_op2_selection: Roulette

prp_HGP_op3_name: OnePointCrossover(internal=0.8)
prp_HGP_op3_probability: 0.10
prp_HGP_op3_selection: Tournament(size=4)
```

Y para el experimenter se han definido los siguientes parámetros:

```
#EXPERIMENTER
prp_experimenter: on
prp_experimenter_repetitions: 10
prp_hypergp: off
prp_generations: 100,200:50
prp_hgp_generations: 10,160:50
```

Según la configuración descrita anteriormente las condiciones de ejecución se traducen en:

- Programación genética en árbol desactivada.
- 4000 individuos en cada población.
- 200 nodos como máximo para cada individuo de la población.
- 50 niveles de profundidad como máximo para cada individuo.
- 10 repeticiones para cada batería de pruebas.

En este caso se ha aumentado considerablemente el número de individuos, de hecho, en las ejecuciones anteriores se han utilizado un número bastante bajo, puesto que Koza, en su publicación habla de poblaciones formadas por 1000 individuos.

También cabe mencionar que se mantiene la limitación máxima de 200 nodos, puesto que permitir que aumente mucho este parámetro “inmuniza” a los individuos respecto a los operadores genéticos.

3.4.1. Resultados

En este caso la programación genética tradicional sí que llega a producir unos relativamente buenos resultados. Sin embargo, el coste computacional de dichos resultados ha sido muy alto.

Por este motivo, se ha limitado el número de repeticiones a una.

En la última generación de la repetición realizada, se obtuvieron los siguientes resultados.

```
-----------------------
|-=  Generation 100 =-|
================================================================================================
     Individual  |         Raw Fitness  |  Adjusted Fit.  | Nodes tree 0 | Depth tree 0 |
------------------------------------------------------------------------------------------------
   Best of Gen.  |             42,0000  |         0,0233  |          101  |            8  |
Generation Mean  |             46,5000  |         0,0211  |    141,0000  |       9,0000  |
  Worst of Gen.  |             47,0000  |         0,0208  |          141  |            9  |
------------------------------------------------------------------------------------------------

-------------------------------------------------------------
     Time (ms.)  |     Population Mean  | Total Population Time  |
-------------------------------------------------------------
  Breeding Time  |              0,1838  |              735,0000  |
Evaluation Time  |           5840,5945  |         23362378,0000  |
================================================================================================
```

Como se puede observar, el mejor individuo de la generación ha conseguido un fitness de 42. Este resultado es mejor que cualquiera de las otras ejecuciones que se han hecho sin programación genética en árbol, que han sido 44 y 51.

Se han extraído los mejores resultados de los 3 experimentos para realizar una comparativa de resultados y coste de obtenerlos. Comenzando por los resultados:

Mejores individuos	Fitness
Experimento 1 con PGA	32
Experimento 2 con PGA	31
Experimento 1 sin PGA	44
Experimento 2 sin PGA	51
Experimento 3 (sin PGA)	42

Tabla 15 - Mejores individuos en los tres experimentos

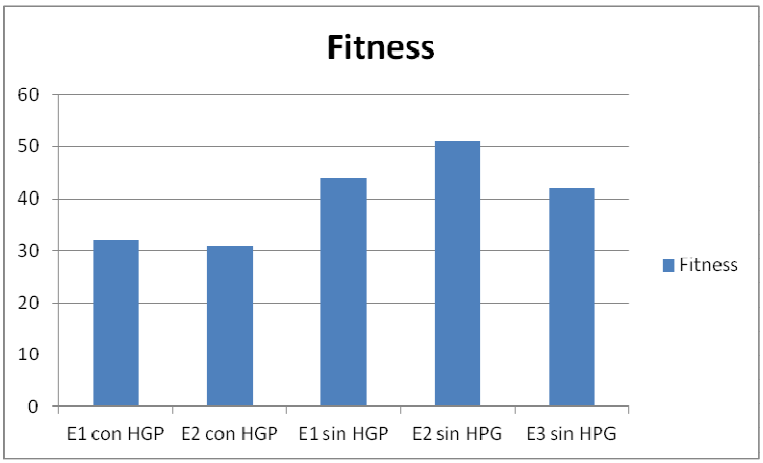


Tabla 16 - Gráfica con los fitness de los mejores individuos

Como se puede observar tanto en la gráfica como en la tabla, teniendo en cuenta que cuanto más próximo a 0, mejor es el resultado de un individuo, los mejores resultados se obtienen, con una diferencia de, aproximadamente, 10 puntos, utilizando la programación genética en árbol. La programación genética tradicional ha mostrado unos resultados más limitados.

También se observa en programación genética tradicional que una única ejecución del experimento 3 ha dado mejores resultados que las 160 ejecuciones que se han realizado para el experimento 1 y las otras tantas realizadas para el experimento 2.

Puesto que del experimento 3 sólo se ha podido realizar una ejecución, ahora procederemos a calcular la misma comparativa, pero teniendo en cuenta los resultados medios y el tiempo medio empleado en encontrar la solución.

Mejores individuos	Fitness
Experimento 1 con PGA	47,26
Experimento 2 con PGA	47,42
Experimento 1 sin PGA	62,82
Experimento 2 sin HPG	63,2
Experimento 3 (sin HPG)	42

Tabla 17 - Fitness medio de los tres experimentos

Utilizando los resultados medios, para poder evaluar los resultados del experimento 3 de una forma un poco más justa, se puede observar que los resultados cambian sensiblemente. El experimento 3 es el que arroja el mejor resultado de todas las ejecuciones, aventajando incluso a las ejecuciones con programación genética en árbol en 5 puntos y en más de 20 puntos respecto a las tradicionales.

La misma gráfica de antes con estos valores quedaría de la forma:

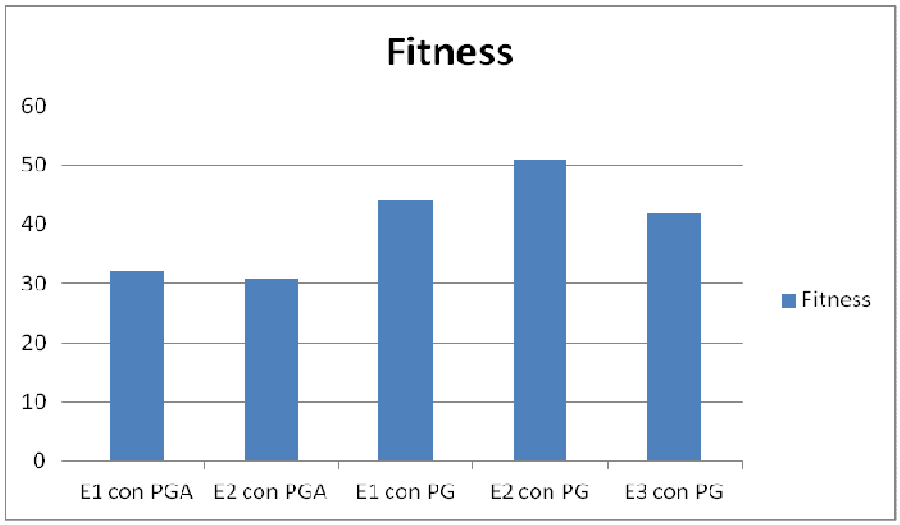


Tabla 18 - Gráfica con los fitness medios de los tres experimentos

Como ya se ha comentado antes, el experimento 3 ha dado unos resultados mejores en media (teniendo en cuenta que del experimento 3 sólo se dispone de una ejecución, por lo que es una media estimada, no real) frente al resto de experimentos.

Esto tiene una explicación muy sencilla, mientras que los dos primeros experimentos contaban únicamente con 600 individuos por generación, el tercer experimento contaba con 4000 individuos, esto es 4000 posibles soluciones evolucionando hacia la mejor y más adaptada solución.

Como se puede intuir, esto conlleva otro factor a tener en cuenta, el tiempo de ejecución de evolucionar y evaluar 600 individuos es proporcionalmente inferior al de tratar con 4000 individuos. En la tabla inferior se pueden consultar los tiempos medios empleados en los experimentos.

Tiempo medio por repetición	Tiempo (seg)
Experimento 1 con PGA	5373,29
Experimento 2 con PGA	4293,29
Experimento 1 sin PGA	1041,54
Experimento 2 sin PGA	990,88
Experimento 3 (sin PGA)	2325845

Tabla 19 - Tiempos medios de ejecución por repetición de los tres experimentos

Como se puede observar el coste de evolucionar individuos arbóreos es mucho más elevado (del orden de 4 ó 5 veces mayor) que el de evolucionar individuos tradicionales, sin embargo, el coste de evolucionar un número mucho mayor de individuos (pasando de 600 a 4000) es muchísimo más alto.

A continuación se ofrece una gráfica para ilustrar estos datos.

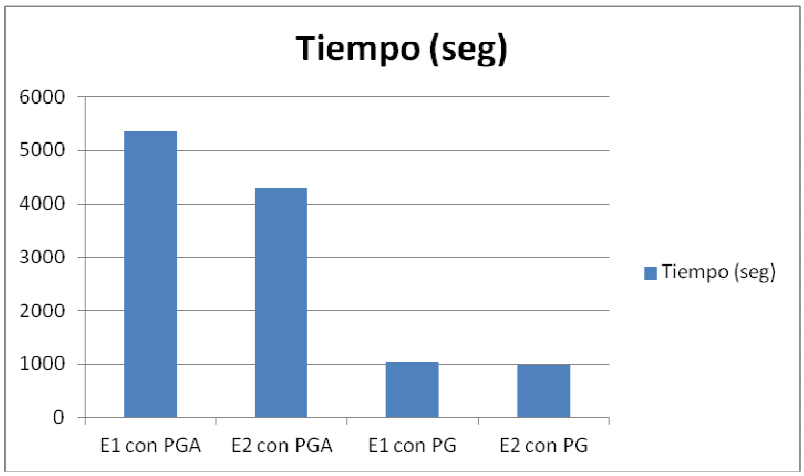


Tabla 20 - Gráfica con los tiempos medios de ejecución de los dos primeros experimentos

Como se ha comentado con anterioridad el tiempo de ejecución de la programación genética en árbol es considerablemente mayor de lo que requiere la programación genética tradicional.

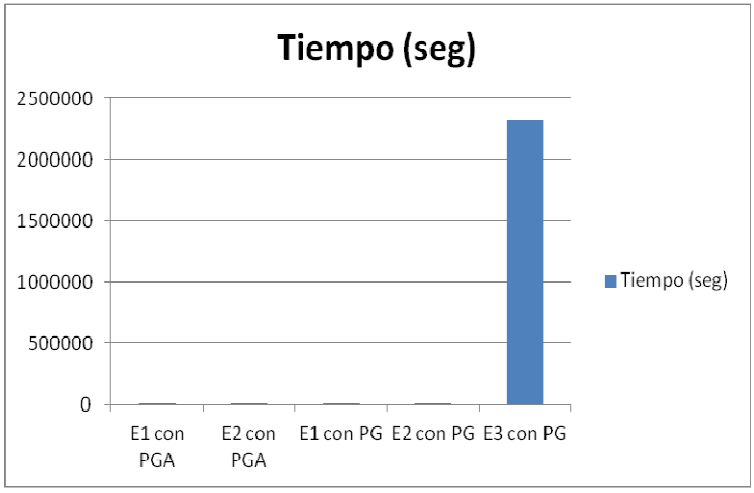


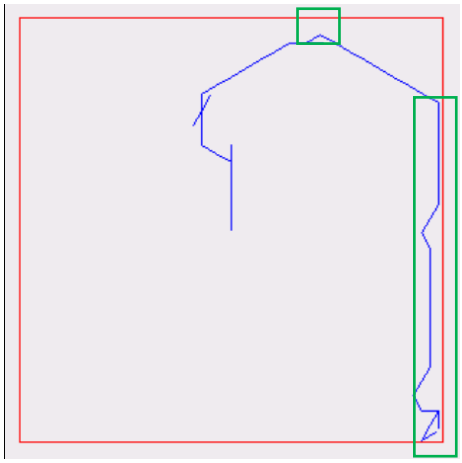
Tabla 21 - Tiempo medio de ejecución de los tres experimentos

Sin embargo esta diferencia resulta intrascendente si tenemos en cuenta el tiempo empleado por el experimento 3. Éste es el motivo por el que no se pudieron realizar más repeticiones para este experimento, con los medios disponibles no ha sido

viable mantener en ejecución un proceso que cada repetición tardaba aproximadamente 27 días.

3.4.2. Visualización de los resultados

En este experimento tan sólo se puede visualizar los resultados de la única repetición cuya ejecución ha terminado.



E3 Mejor Robot sin HGP.avi

Ilustración 43 - Mejor individuo del experimento 3

No es un mal resultado teniendo en cuenta que esta batería se hubiese compuesto de 10 repeticiones con ejecuciones como esta, por lo que no hubiese sido difícil que alguna de las otras ejecuciones hubiese mejorado los resultados. Por este motivo, numéricamente se ha comparado con las ejecuciones medias de los otros experimentos, pero visualmente no se puede comparar.

3.4.3. Conclusiones

Finalmente, aunque de este experimento 3 sólo se puede hablar a partir de estimaciones, se puede observar que el coste de desarrollar un individuo que resulte competitivo con los individuos arbóreos resulta excesivamente alto.

Aunque el resultado es bueno, como se ha visto anteriormente:

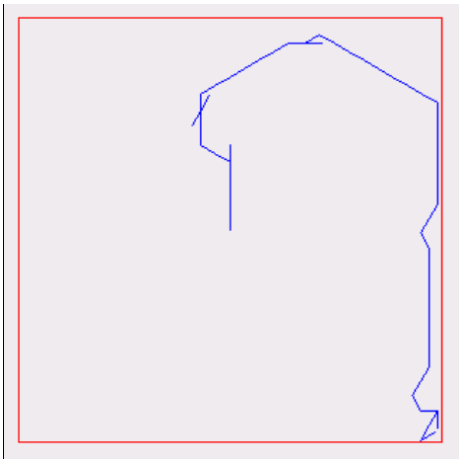


Ilustración 44 - Individuo del experimento 3

Se puede observar que el robot, desde el centro de la habitación, busca la pared y una vez que la encuentra comienza a seguirla. En esto es comparable con la mejor de las otras soluciones:

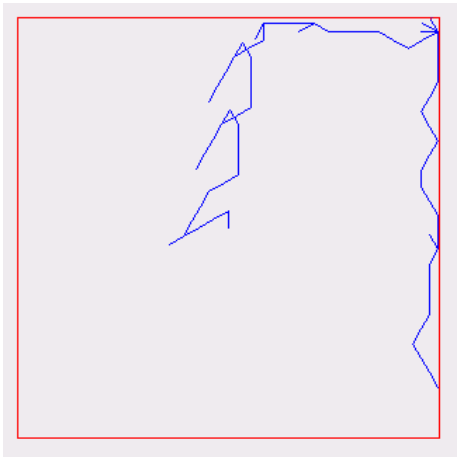


Ilustración 45 - Mejor individuo de todos los experimentos

Sin embargo, para encontrar esta solución, que aún así es peor, ha tardado 2325845 segundos, mientras que la mejor ejecución de todas, la que se muestra, tardó

1386 segundos. Lo que hace un tiempo de ejecución del 167809,88% respecto la ejecución que utilizó la programación genética en árbol.

De hecho si comparamos el tiempo de ejecución de esta repetición del experimento con todas las ejecuciones del primero o segundo experimento con programación genética en árbol, que son la que han dado unas ejecuciones más similares, obtenemos la siguiente tabla.

Tiempo repeticiones	Tiempo (seg)	Tiempo (horas)
Experimento 1 con PGA (160)	859726	238,81
Experimento 2 con PGA (160)	686927	190,81
Experimento 3 (1)	2325845	646,07

Tabla 22 - Tiempo requerido por los experimentos

Como se puede observar tarda más la ejecución de una única repetición del experimento 3 que las 160 repeticiones con programación genética en árbol activada que realizan los otros dos experimentos.

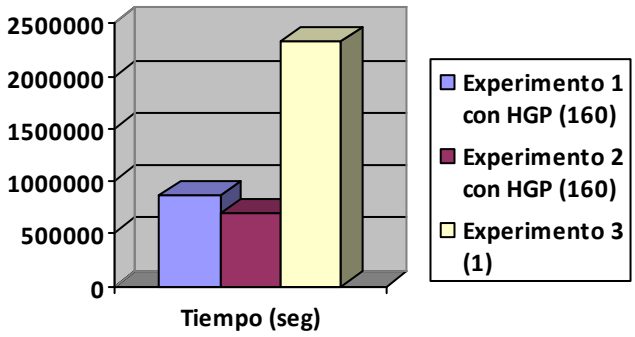


Tabla 23 - Gráfica de los tiempos requeridos por los experimentos

De esto se deduce que la programación genética en árbol, en igualdad de condiciones ofrece mejores resultados y de forma más eficiente que la programación genética tradicional, como mínimo para problemas como este.

4. Conclusiones del estudio

Una vez completados los análisis de los tres casos de ejecución procederemos a extraer las conclusiones generales que pretendían extraerse con este estudio.

En este punto realizaremos un compendio de los resultados obtenidos por la programación genética tradicional, los resultados obtenidos por la programación genética y finalmente ofreceremos las conclusiones en cuya búsqueda se ha realizado este estudio: la conveniencia o inconveniencia de utilizar la programación genética en árbol como medio de acelerar la programación genética tradicional.

Por último, comentaremos los resultados obtenidos, razones por las que los resultados han sido mejores y peores y las cosas que han influido para ello.

4.1. Programación Genética

En este apartado comentaremos los resultados obtenidos por la programación genética tradicional a lo largo de los 3 experimentos en los que se ha ejecutado.

Resultados Experimento 1

Experimento	Average Fitness	Best Fitness
exp_1	62,900	57,000
exp_3	62,700	58,000
exp_5	64,200	57,000
exp_7	63,100	61,000
exp_9	62,700	60,000
exp_11	62,300	51,000
exp_13	63,400	58,000
exp_15	63,200	52,000
exp_17	64,000	58,000
exp_19	62,900	53,000
exp_21	59,600	44,000
exp_23	63,400	61,000
exp_25	62,600	59,000
exp_27	64,100	57,000
exp_29	61,900	58,000
exp_31	62,100	59,000

Resultados Experimento 2

Experimento	Average Fitness	Best Fitness
exp_1	64,600	60,000
exp_3	61,400	55,000
exp_5	63,900	59,000
exp_7	62,600	52,000
exp_9	63,400	56,000
exp_11	64,500	62,000
exp_13	61,500	54,000
exp_15	63,600	57,000
exp_17	63,800	56,000
exp_19	64,900	61,000
exp_21	61,500	51,000
exp_23	61,600	57,000
exp_25	61,600	57,000
exp_27	63,400	55,000
exp_29	64,000	55,000
exp_31	64,900	59,000

Tabla 24 - Resultados de la programación genética

Como se puede observar en los experimentos 1 y 2, los resultados obtenidos no son concluyentes.

El mejor de todos los fitness para el primer experimento es 44, en una ejecución que se puede considerar afortunada, porque si observamos las medias todas ellas andan en torno al 60, por lo que se podría hablar de él incluso como un dato atípico.

El mejor de todos los fitness para el segundo experimento es 51, manteniendo las medias por encima de 60. Estos resultados no son particularmente buenos.

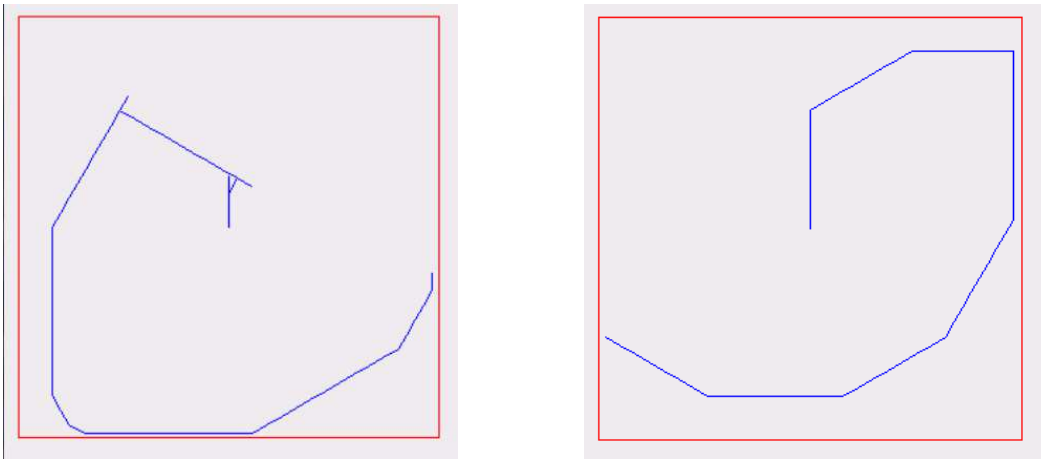


Ilustración 46 - Recorridos de los mejores resultados de la programación genética

Como se puede observar en ambos resultados, realizan el recorrido de un segmento de pared, sin embargo es un recorrido bastante pequeño, sin grandes aproximaciones a ninguna otra pared.

Esto se puede deber a varios factores.

- Los operadores y terminales no son adecuados para tratar de resolver este problema.
- Los individuos son demasiado simples o la complejidad del problema es demasiado alta.
- La función de fitness no es capaz de guiar a los individuos hacia la solución.

Como veremos más adelante, los operadores y terminales sí que son adecuados, porque a través de la programación genética en árbol se obtienen resultados bastante mejores que los obtenidos mediante la programación genética tradicional.

Es un hecho que el problema es de una gran complejidad, y los individuos requieren un gran tamaño para recorrer la habitación. De hecho esta es la razón por la que no la recorren por completo, puesto que ya hemos visto que ejecutando los individuos el doble de veces, se consiguen resultados bastante más interesantes:

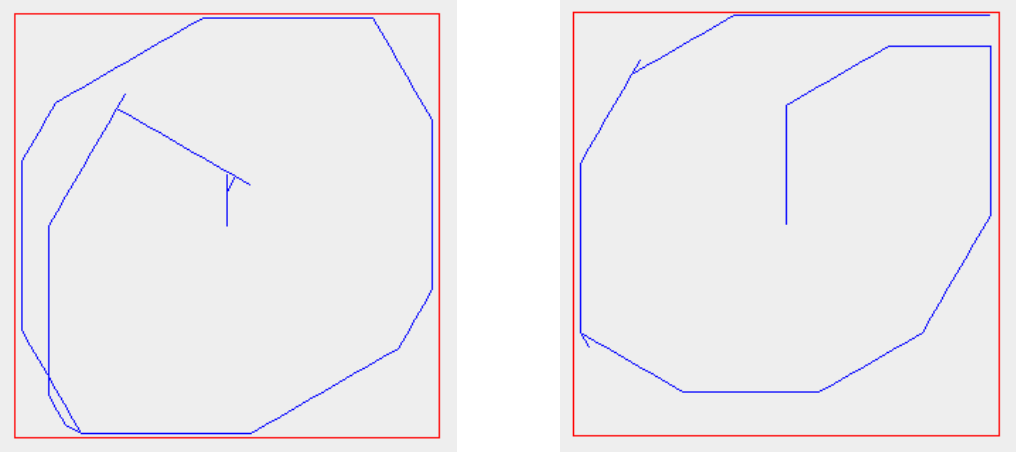


Ilustración 47 – Soluciones de la programación genética ejecutadas el doble de veces

Por último, respecto a la función de fitness, también se puede descartar que sea un problema, puesto que como se acaba de ver, simplemente duplicando las veces que se ejecuta cada individuo, la solución mejora considerablemente.

En cuanto a programación genética tradicional, por otro lado tenemos el tercer experimento.

Este experimento trataba de confirmar que la complejidad del problema fuese demasiado alta y no problema de los individuos. Para ello lanzamos una ejecución con 4000 individuos por generación. A través de un coste computacional exageradamente elevado, obtenemos unos resultados sorprendentemente buenos, lo que sugiere que la programación genética es perfectamente capaz de resolver este problema, salvando la propia complejidad del problema.

Resultados Experimento 3

Experimento	Average Fitness	Best Fitness
exp_1	42,000	42,000

Cabe destacar de este experimento la tasa media de fitness, muy inferior a la obtenida en cualquier otro experimento. Hay que tener en cuenta que es una tasa media estimada, puesto que tan sólo se ha realizado una repetición. Sin embargo, aunque podría ser un resultado atípico, la propia definición de resultado atípico da validez a la estimación puesto que sería teóricamente difícil que realizando un experimento casualmente, de un resultado atípico.

Por la falta de confiabilidad, he mantenido este experimento aparte, aunque no excluido, puesto que muestra un comportamiento bastante más acertado a una solución correcta que los anteriores.

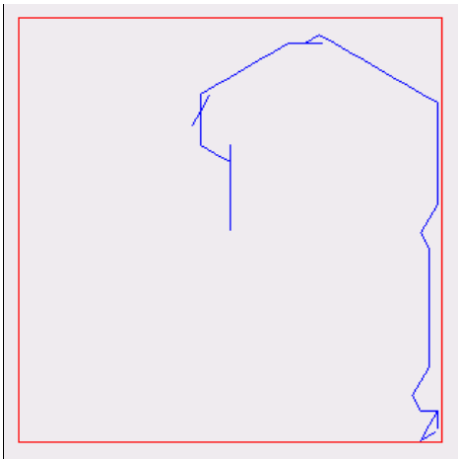


Ilustración 48 - Mejor resultado de la programación genética

Como se puede observar, el robot, una vez se acerca a una pared es capaz de recorrerla entera, mientras que en los robots de los experimentos anteriores, este recorría un trozo de pared, pero antes de terminarla se alejaba de ella dando un resultado menos óptimo.

4.2. Programación genética en árbol

En este apartado comentaremos los resultados obtenidos por la programación genética en árbol a lo largo de los 2 experimentos en los que se ha ejecutado.

Resultados Experimento 1

Experimento	Average Fitness	Best Fitness
exp_0	52,000	44,000
exp_2	53,800	45,000
exp_4	54,600	46,000
exp_6	57,600	49,000
exp_8	44,800	41,000
exp_10	46,500	35,000
exp_12	45,900	42,000
exp_14	44,200	37,000
exp_16	44,700	38,000
exp_18	43,100	37,000
exp_20	43,100	32,000
exp_22	45,000	41,000
exp_24	43,100	37,000
exp_26	46,200	36,000
exp_28	46,400	43,000
exp_30	45,200	40,000

Resultados Experimento 2

Experimento	Average Fitness	Best Fitness
exp_0	55,900	49,000
exp_2	51,500	31,000
exp_4	55,700	49,000
exp_6	52,400	34,000
exp_8	46,700	41,000
exp_10	44,100	39,000
exp_12	43,500	37,000
exp_14	46,000	43,000
exp_16	44,700	40,000
exp_18	44,700	37,000
exp_20	46,500	42,000
exp_22	46,400	36,000
exp_24	44,900	39,000
exp_26	47,100	40,000
exp_28	45,100	36,000
exp_30	43,500	40,000

Tabla 25 - Resultados de la programación genética en árbol

Como se puede observar, los resultados obtenidos en las ejecuciones con la programación genética en árbol han sido bastante buenos.

Las medias en ambos experimentos se encuentran en torno a los 45 puntos, siendo las mejores ejecuciones de 32 y 31 puntos para los experimentos uno y dos respectivamente.

Observando las representaciones gráficas de los resultados, podremos comprobar que los individuos con programación genética en árbol son bastante metódicos, buscando una pared y recorriéndola en cuanto llegan a ella.

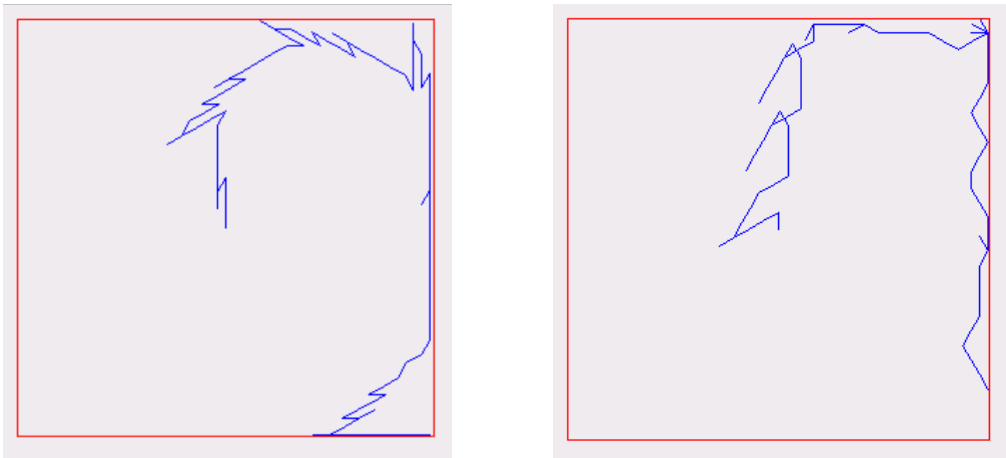


Ilustración 49 - Recorridos de los mejores resultados de la programación genética en árbol

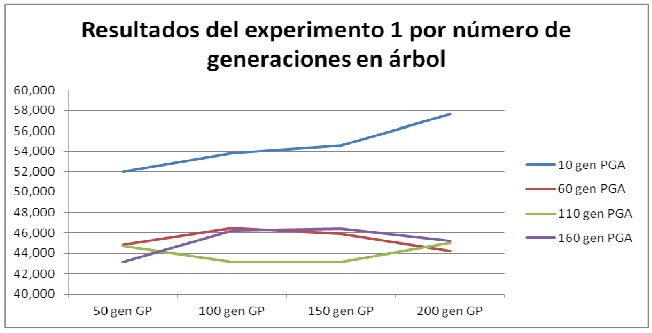
Resulta especialmente llamativo en estos casos, como ya se ha comentado, la meticulosidad de las soluciones. Una vez que un robot ha conseguido encontrar una pared se aferra a ella, no trata de cruzar la habitación por en medio.

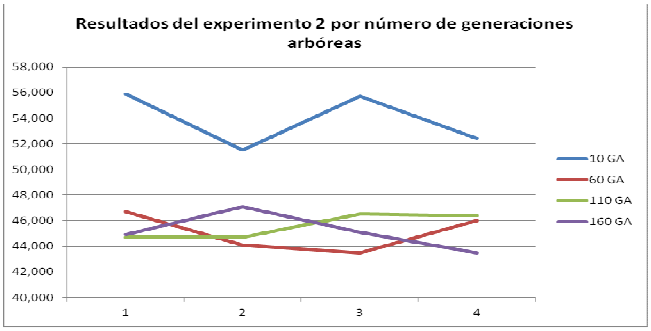
Otro aspecto que resulta clave y que demuestra la capacidad de los individuos es el hecho de que puedan girar en las esquinas.

Como se puede observar, ambos individuos son capaces, con mayor o menor grado de éxito, de girar en las esquinas de la habitación y continuar avanzando pegados a las paredes, sin separarse de ellas, ni perderlas.

En este sentido la única objeción que se le podría poner al primer individuo es que recorre la última pared en sentido contrario, en lugar de avanzando, pero dada la función de fitness que se utiliza, cumple perfectamente con lo que se espera de él. Únicamente habría supuesto una penalización en fitness si hubiese podido seguir recorriendo la habitación.

Otro detalle que se debe comentar es que tampoco hace falta un número demasiado alto de generaciones arbóreas para obtener unos buenos resultados, como han confirmado los dos primeros experimentos realizados:





Como se muestra en las gráficas, 10 ha sido un número insuficiente de generaciones arbóreas, sin embargo 110 o 160 no han reportado mejores respecto a las ejecuciones con 60 generaciones arbóreas.

Esto se puede observar en los dos experimentos, teniendo en cuenta que cada punto de la gráfica representa el fitness medio de 10 repeticiones, lo que permite concluir que la programación genética en árbol permite mejorar de forma radical la efectividad de la programación genética, como mínimo en este problema y los que tengan características similares a este.

4.3. Programación Genética vs Programación genética en árbol

Una vez hemos revisado los resultados de la programación genética y la arbórea procederemos a comprarlos de forma directa.

Los resultados numéricos son contundentes. Excepto el experimento 3, que en media podría ofrecer unos resultados mejores que los de la programación genética en árbol, estos en principio quedarían eliminados por el coste computacional.

Mejores resultados

Ejecución	Average Fitness	Best Fitness
Exp1 con PGA	43,100	32,000
Exp2 con PGA	51,500	31,000
Exp1 sin PGA	59,600	44,000
Exp2 sin PGA	61,500	51,000
Exp3 sin PGA	42,000	42,000

Tabla 26 - Resultados de todos los experimentos

La representación en gráfica de esta tabla sería de la forma:

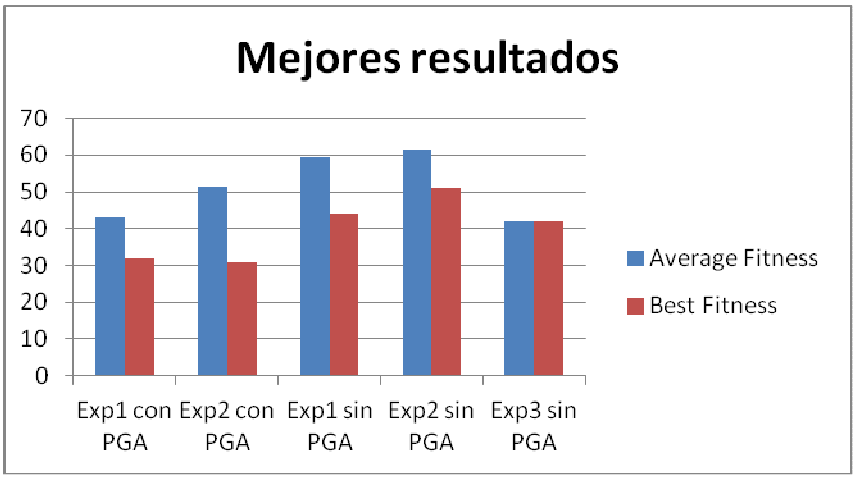


Tabla 27 - Gráfica de los resultados de todos los experimentos

Con estos datos se puede observar como la programación genética en árbol ha proporcionado mejores resultados que la programación genética tradicional, siendo incluso comparable la tasa media de fitness del Experimento 1 con la del Experimento 3, cuando el coste computacional de este último es enormemente mayor que el del Experimento 1.

Esto ya se pudo observar en las gráficas de los resultados de los experimentos anteriores:

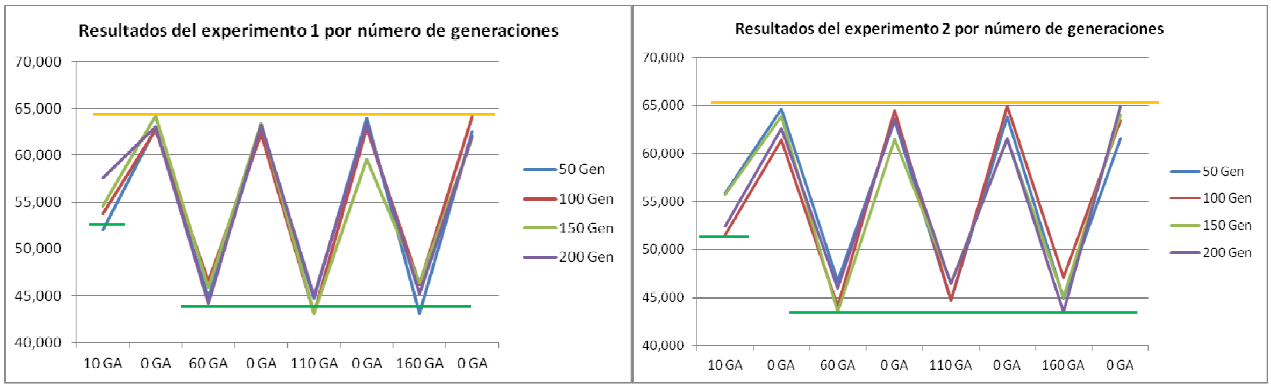


Tabla 27 - Resultados de los dos primeros experimentos

En estas gráficas se puede comprobar cómo los resultados de la programación genética (línea amarilla) son claramente peores que los de la programación genética en árbol (línea verde), observándose únicamente una aproximación en el caso de las 10 generaciones arbóreas.

Teniendo en cuenta el experimento 3, en las gráficas anteriores nos daría la línea azul:

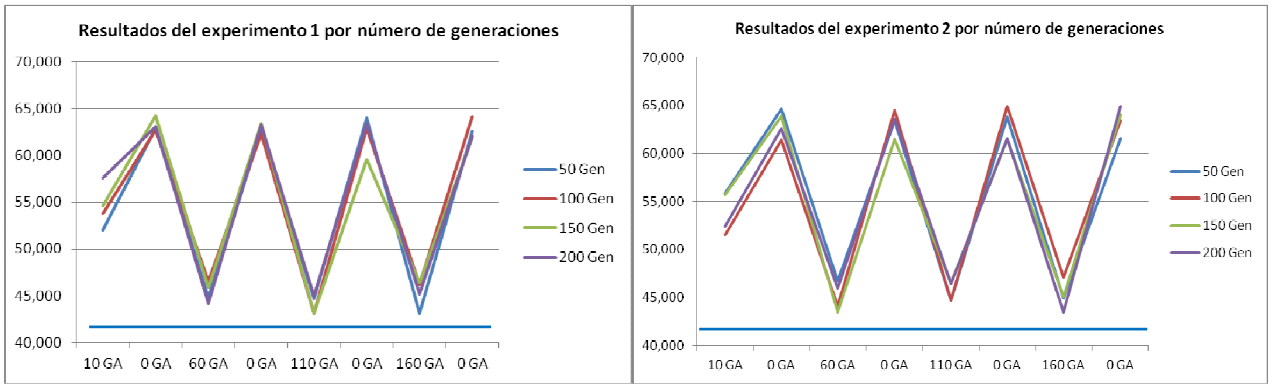


Tabla 28 - Resultados de los tres experimentos

Sin embargo, hemos de recordar que para obtener dicho resultado, ligeramente mejor que las ejecuciones con 60 o más generaciones arbóreas, se ha requerido un tiempo de ejecución no admisible:

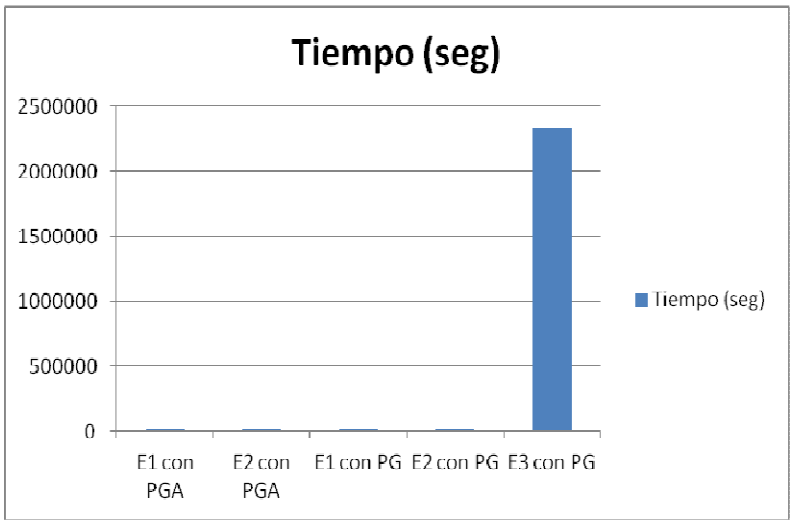


Tabla 29 - Tiempo empleado en resolver los experimentos

Obviando, además, el hecho de que se ha realizado una única repetición de dicho experimento.

Con estos resultados se puede obtener las siguientes conclusiones:

1. **La programación genética en árbol potencia la programación genética tradicional, permitiendo obtener mejores resultados en igualdad de condiciones.**

2. **La programación genética en árbol permite obtener resultados similares a otras ejecuciones de programación genética tradicional que han requerido una enorme cantidad de recursos para obtener dichos resultados.**

Además, también se puede concluir que para este problema, **la estructura arbórea de los individuos proporciona información acerca de lo buena o mala que es una solución**, independientemente de los operadores que conformen dicha estructura arbórea.

Esto es intuitivo, puesto que un robot que necesite recorrer una pared, necesariamente tendrá que concatenar numerosos operadores de avanzar, dando lugar a árboles de estructura característica.

Por último quería comentar que los experimentos se han realizado con los medios disponibles, esto es un PC no muy potente con un procesador Pentium 4, no sobre un cluster de varias máquinas en paralelo. Lo que puede influir si se tienen en cuenta otros estudios realizados en entornos de mayor potencia.

Anexos

Anexo 1: Evolution of Subsumption Using Genetic Programming

John R. Koza
Computer Science Department
Stanford University
Stanford, CA 94305 USA
E-MAIL: Koza@Sunburn.Stanford.Edu
PHONE: 415-941-0336 FAX: 415-941-9430

The recently developed genetic programming paradigm is used to evolve emergent wall following behavior for an autonomous mobile robot using the subsumption architecture.

1. INTRODUCTION AND OVERVIEW

The repetitive application of seemingly simple rules can lead to complex overall emergent behavior. Emergent functionality means that overall functionality is not achieved in the conventional tightly coupled, centrally controlled way, but, instead, indirectly by the interaction of relatively primitive components with the world and among themselves [Steels 1991]. Emergent functionality is one of the main themes of research in artificial life [Langton 1989].

In this paper, we use the genetic programming paradigm to evolve a computer program that exhibits emergent behavior and enables an autonomous mobile robot to follow the walls of an irregularly shaped room. The evolutionary process is driven only by the fitness of the programs in solving the problem.

2. BACKGROUND ON GENETIC ALGORITHMS

John Holland's pioneering 1975 *Adaptation in Natural and Artificial Systems* described how the evolutionary process in nature can be applied to artificial systems using the genetic algorithm operating on fixed length character strings [Holland 1975].

Holland demonstrated that a population of fixed length character strings (each representing a proposed solution to a problem) can be genetically bred using the Darwinian operation of fitness proportionate reproduction and the genetic operation of recombination. The recombination operation combines parts of two chromosomelike fixed length character strings, each selected on the basis of their fitness, to produce new offspring strings.

Current work in the field of genetic algorithms is reviewed in Goldberg [1989], Belew and Booker [1991], Davis [1987, 1991], Rawlins [1991] and Meyer and Wilson [1991].

3 . BACKGROUND ON GENETIC PROGRAMMING

For many problems, the most natural representation for solutions are computer programs whose size, shape, and content have not been determined in advance. It is unnatural and difficult to represent computer programs of dynamically varying size and shape with fixed length character strings.

Although one might think that computer programs are so epistatic that they could only be genetically bred in a few especially congenial problem domains, we have shown that computer programs can be genetically bred to solve a surprising variety of problems in many different areas [Koza 1992], including

- emergent behavior (e.g. discovering a computer program which, when executed by all the ants in an ant colony, enables the ants to locate food, pick it up, carry it to the nest, and drop pheromones along the way so as to produce cooperative emergent behavior) [Koza 1991a],
- planning (e.g. navigating an artificial ant along an irregular trail) [Koza 1990b],
- finding minimax strategies for games (e.g. differential pursuer-evader games; discrete games in extensive form) by both evolution and co-evolution [Koza 1991b],
- optimal control (e.g. centering a cart and balancing a broom in minimal time by applying a bang-bang force to the cart) (Koza and Keane 1990a, 1990b],
- machine learning of functions (e.g. learning the Boolean 11-multiplexer function) [Koza 1991d], generation of random numbers (using entropy as fitness) [Koza 1991c],
- symbolic regression, integration, differentiation, and symbolic solution to general functional equations for a solution in the form of a function (including differential equations with initial conditions, and integral equations) [Koza 1990], and
- simultaneous architectural design and training of neural nets [Koza and Rice 1991a].

A videotape visualization of the application of genetic programming to planning, emergent behavior, empirical discovery, inverse kinematics, and game playing can be found in the *Artificial Life II Video Proceedings* [Koza and Rice 1991b].

3.1. OBJECTS IN GENETIC PROGRAMMING

In genetic programming, the individuals in the population are compositions of functions and terminals appropriate to the particular problem domain. The set of functions used typically includes arithmetic operations, mathematical functions, conditional logical operations, and domain-specific functions. The set of terminals used typically includes inputs (sensors) appropriate to the problem domain and possibly various constants. Each function in the function set should be well defined for any

combination of elements from the range of every function that it may encounter and every terminal that it may encounter.

One can now view the search for a solution to the problem as a search in the hyperspace of all possible compositions of functions and terminals (i.e. computer programs) that can be recursively composed of the available functions and terminals. The symbolic expressions (S-expressions) of the LISP programming language are an especially convenient way to create and manipulate the compositions of functions and terminals described above. These S-expressions in LISP correspond directly to the parse tree that is internally created by most compilers.

3 . 2 . OPERATIONS IN GENETIC PROGRAMMING

The basic genetic operations for the genetic programming paradigm are reproduction (e.g. fitness proportionate reproduction) and crossover (recombination).

The reproduction operation copies an individual in the population into the new population for the next generation.

The crossover (recombination) operation is a sexual operation that operates on two parental LISP S-expressions and produces two offspring S-expressions using parts of each parent. The crossover operation creates new offspring Sexpressions by exchanging sub-trees (i.e. sublists) between the two parents. Because entire sub-trees are swapped, this crossover operation always produces syntactically and semantically valid LISP S-expressions as offspring regardless of the crossover points.

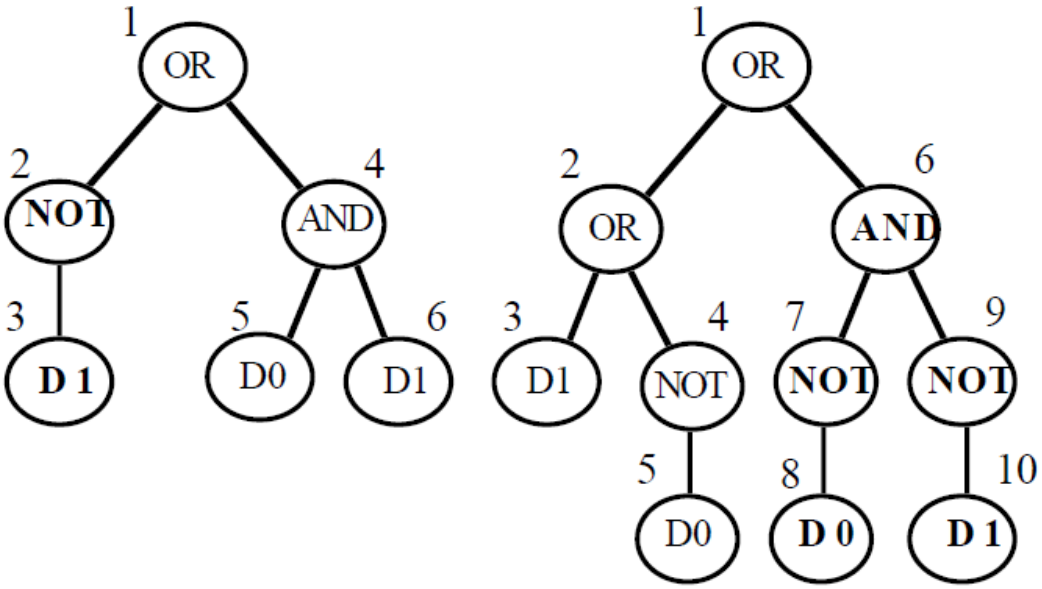


Figure 1: Two parental computer programs shown as trees with ordered branches. Internal points of the tree correspond to functions (i.e. operations) and external points correspond to terminals (i.e. input data).

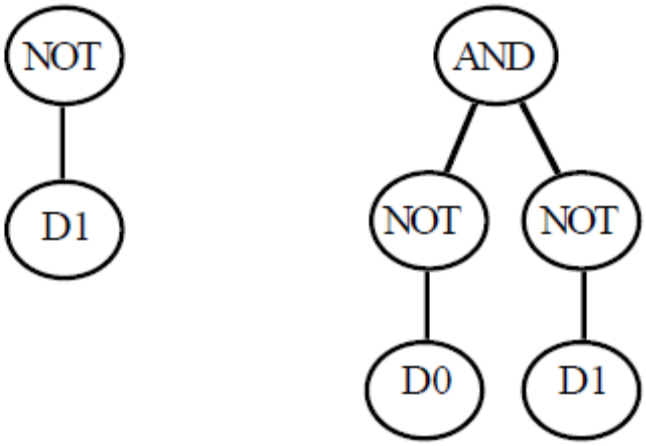


Figure 2: The two crossover fragments

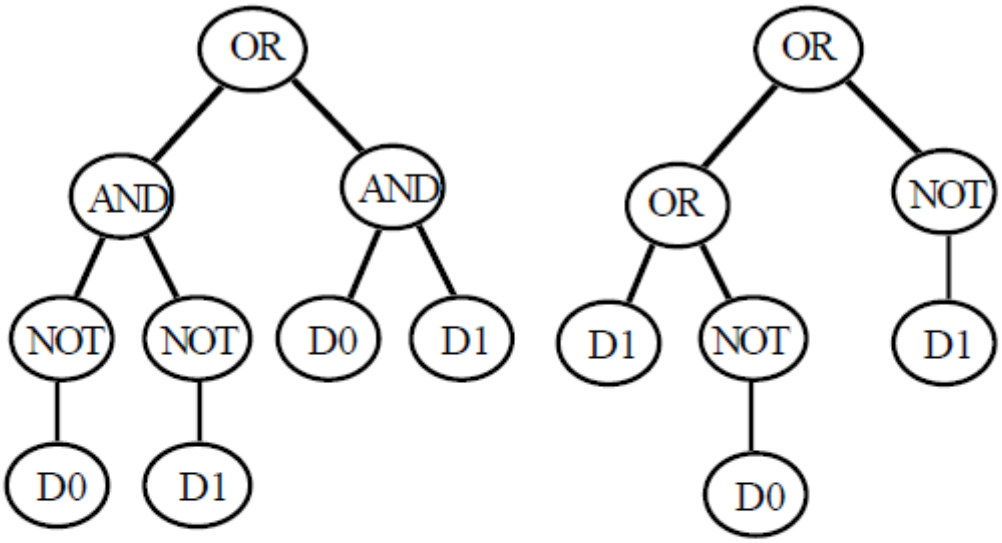


Figure 3: Offspring resulting from crossover

For example, consider the two parental S-expressions:

```
(OR (NOT D1) (AND D0 D1))

(OR (OR D1 (NOT D0))
    (AND (NOT D0) (NOT D1)))
```

Figure 1 graphically depicts these two S-expressions as rooted, point-labeled trees with ordered branches. The numbers on the points of the tree are for reference only.

Assume that the points of both trees are numbered in a depth-first way starting at the left. Suppose that point no. 2 (out of 6 points of the first parent) is randomly selected as the crossover point for the first parent and that point no. 6 (out of 10 points of the second parent) is randomly selected as the crossover point of the second parent. The crossover points in the trees above are therefore the NOT in the first parent and the AND in the second parent.

Figure 2 shows the two crossover fragments are two sub-trees. These two crossover fragments correspond to the bold subexpressions (sub-lists) in the two parental LISP S-expressions shown above.

Figure 3 shows the two offspring resulting from the crossover.

Note that the first offspring in Figure 3 is an S-expression for the Boolean even-parity (i.e. equal) function, namely
`(OR (AND (NOT D0) (NOT D1)) (AND D0 D1)).`

3 . 3 . EXECUTION OF GENETIC PROGRAMMING

The genetic programming paradigm, like the conventional genetic algorithm, is a domain independent method. It proceeds by genetically breeding populations of computer programs to solve problems by executing the following three steps:

(1) Generate an initial population of random compositions of the functions and terminals of the problem (computer programs).
(2) Iteratively perform the following sub-steps until the termination criterion has been satisfied:
    a. Execute each program in the population and assign it a fitness value according to how well it solves the problem.
    b. Create a new population of computer programs by applying the following two primary operations. The operations are applied to computer program(s) in the population chosen with a probability based on fitness.
        i. *Reproduction*: Copy existing computer programs to the new population.
        ii. *Crossover*: Create two new computer programs by genetically recombining randomly chosen parts of two existing programs.
(3) The single best computer program in the population at the time of termination is designated as the result of the genetic programming paradigm. This result may be a solution (or approximate solution) to the problem.

4 . THE WALL FOLLOWING PROBLEM

Mataric [1990] described the problem of controlling an autonomous mobile robot to perform the task of following the walls of an irregular room.

The robot is capable of executing the following five primitive motor functions: moving forward by a constant distance, moving backward by a constant distance, turning right by 30 degrees, turning left by 30 degrees, and stopping.

The robot has 12 sonar sensors which report the distance to the nearest wall. Each sonar sensor covers a 30 degree sector around the robot. In addition, there was a sensor for the STOPPED condition of the robot.

Figure 4 shows an irregularly shaped room and the distances reported by the 12 sonar sensors. The robot is shown at point (12, 16) near the center of the room. The north (top) wall and west (left) wall are each 27.6 feet long.

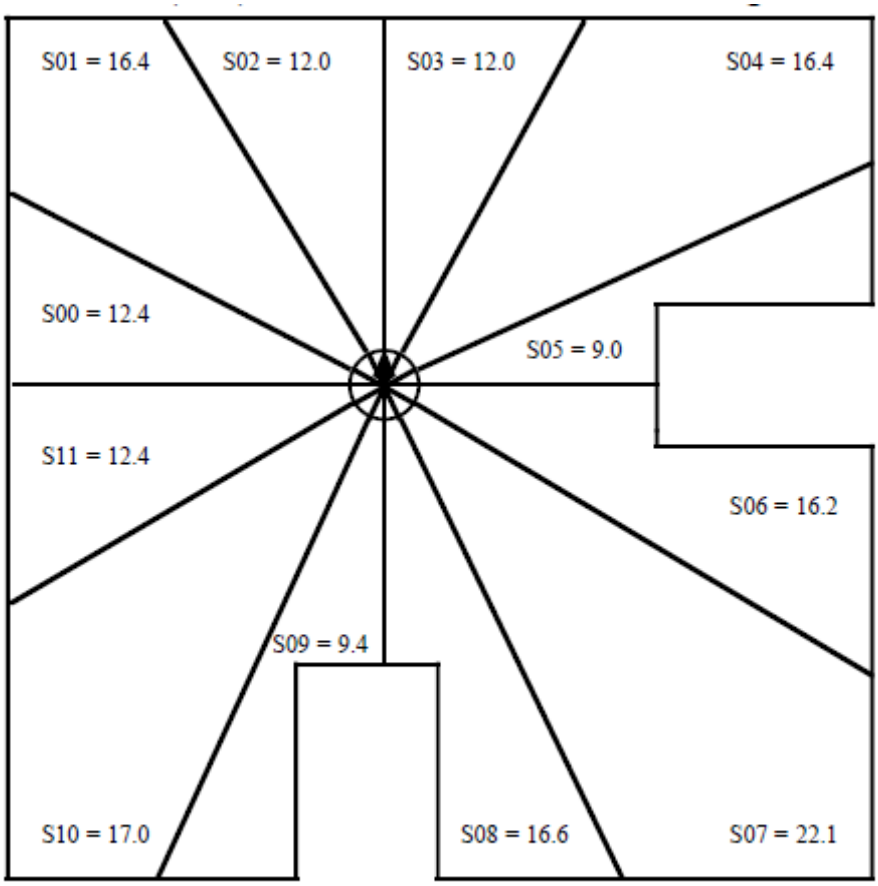


Figure 4: Irregular room with robot with 12 sonar sensors located near middle of the room

One can envision solving the wall following problem in one of three approaches, namely

(1) the conventional approach to building control systems for for autonomous mobile robots,
(2) the subsumption architecture, and
(3) genetic programming.

Regardless of which of these three approaches is used to solve the wall following problem, the 13 sensors can be viewed as the input to an as-yet unwritten computer program in a style appropriate to the three approaches. This as-yet unwritten program will process these inputs and will cause the activation, in some order, of the various primitive motor functions. This as-yet-unwritten program will be a composition of the five available primitive motor functions and the 13 available sensors.

Note that the 13 sensors and five primitive motor functions are the starting point of all three approaches. They are given as part of the statement of the problem.

5 . THE CONVENTIONAL ROBOTIC APPROACH

The conventional approach to building control systems for autonomous mobile robots is to decompose the overall problem into a series of functional units that perform functions such as perception, modeling, planning, task execution, and motor control. A central control system then executes each functional unit in this decomposition and then passes the results on to the next functional unit in an orderly, closely coupled, and synchronized manner. For example, the perception unit senses the world and the results of this sensing are then passed to a modeling module which attempts to build an internal model of the perceived world. The internal model resulting from this modeling is then passed on to a planning unit which computes a plan. The plan might be devised by a consistent and logically sound technique (e.g. resolution and unification) or it might be devised by one of the many heuristic techniques of symbolic artificial intelligence. In any event, the resulting plan is passed on to the task execution unit which then executes the plan by calling on the motor control unit. The motor control unit then acts directly on the external world.

In this conventional approach, only a few of the functional units (e.g. the perception unit and motor control unit) typically are in direct communication with the world. All functional units must typically be executed in their intended orderly, closely coupled, and synchronized manner in order to make the robot do anything.

6 . THE SUBSUMPTION ARCHITECTURE

The subsumption architecture decomposes the problem into a set of asynchronous task achieving behaviors [Brooks 1986, 1989]. The task achieving behaviors for an autonomous mobile robot might include behaviors such as avoiding objects, wandering, exploring, identifying objects, building maps, planning changes to the world, monitoring changes, and reasoning about behavior of the objects. The task achieving behaviors operate locally and asychronously and are only loosely coupled to one another. In contrast to the conventional approach, each of the task achieving behaviors is typically in direct communication with the world (and each other). The task achieving behaviors in the subsumption architecture are typically much more primitive than the functional units of the conventional approach.

In the subsumption architecture, various subsets of the task achieving behaviors typically exhibit some partial competence in solving a simpler version of the overall problem. Thus, the solution to a more complex version of a problem can potentially be built up by incrementally adding new independent acting parts to existing parts. In addition, the system may be fault tolerant in the sense that the failure of one part does not cause complete failure, but, instead, causes a gracefully degradation of performance to some lower level. In contrast, in the conventional approach, the various functional units have no functionality when operating separately and there is a complete suspension of all performance when one functional unit fails.

In the subsumption architecture, the task achieving behaviors each consist of an applicability predicate, a gate, and a behavioral action. If the current environment satisfies the applicability predicate of a particular behavior, the gate allows the output of

the behavioral action to feed out onto the output line of that behavior. Potential conflicts among behavioral actions are resolved by a hierarchical arrangement of suppressor nodes. As a simple example, suppose that there are three task achieving behaviors with strictly decreasing priority. The applicability predicates and the suppressor nodes of these three behaviors are equivalent to the following composition of ordinary IF conditional functions:

```
(IF A-P-1 BEHAVIOR-1
      (IF A-P-2 BEHAVIOR-2
            (IF A-P-3 BEHAVIOR-3)
```

In particular, if the first applicability predicate (AP-1) is satisfied, then BEHAVIOR-1 is executed. Otherwise, if A-P-2 is satisfied, BEHAVIOR-2 is executed. Otherwise, the lowest priority behavior (i.e. BEHAVIOR-3) is executed.

Mataric (1990) has implemented the subsumption architecture for controlling an autonomous mobile robot by conceiving and writing a set of four programs for performing four task achieving behaviors. The four behaviors together enable a mobile robot called TOTO to follow the walls in an irregular room.

Starting with the five primitive motor functions and the 13 sensors that are part of the definition of the problem, Mataric applied her intelligence and ingenuity and conceived of a set of four task achieving behaviors which together enable a mobile robot to follow the walls in an irregular room. As a matter of preference, Mataric specifically selected her four task achieving behaviors so that their applicability predicates were mutually exclusive (thus eliminating the need for a conflict resolution architecture allowing one task achieving behavior to suppress the behavior of another).

Mataric then wrote a set of four LISP programs for performing the four task achieving behaviors. Mataric's four LISP programs corresponded to the four task achieving behaviors and were called STROLL, AVOID, ALIGN, and CORRECT. Each of these four task achieving behaviors interacted directly with the world and each other.

Various subsets of Mataric's four behaviors exhibited some partial competence in solving part of the overall problem. For example, the robot became capable of collission free wandering with only the STROLL and AVOID behaviors. The robot became capable of tracing convex boundaries with only the addition of only the ALIGN behavior to these first two behaviors. Finally, the robot became capable of general boundary tracing with the further addition of the CORRECT behavioral unit.

Mataric's four LISP programs included nine LISP functions (namely, COND, AND, NOT, IF, >, >=, =, <=, and >). In addition, her four programs internally made use of three constant parameters (defining an edging distance EDG, minimum safe distance MSD, and danger zone DZ), the minimum of all 12 sonar distances (called "Shortest Sonar" or SS), and eight other internally defined variables representing the minimum of various thoughtfully chosen subsets of the 12 sonar distances (e.g. the dynamically computed minimum of a particular three forward facing sensors).

In total, Mataric's four LISP programs consisted a composition of 151 functions and terminals.

The fact that Mataric was able to write four programs enabling an autonomous robot to perform the task of following the wall of an irregular room is evidence (based on this particular problem) for one of the claims of the subsumption architecture, namely, that it is possible to build a control system for an autonomous mobile robot using loosely coupled, asynchronous task achieving behaviors.

Note that if Mataric had wanted to write a computer program for wall following using conventional coupled synchronous robotic techniques, her program would have taken in the same 13 sensors as inputs and caused the activation, in some order, of the same five primitive motor functions as the output of the program.

The conception and design of suitable task achieving behaviors for the subsumption architecture requires considerable ingenuity and skill on the part of the human programmer.

7 . APPLICATION OF GENETIC PROGRAMMING TO THE WALL FOLLOWING PROBLEM

The question arises as to whether an autonomous mobile robot can learn to perform wall following in an evolutionary way, and, in particular, by using genetic programming. This learning would include learning both the necessary task achieving behaviors (including the applicability predicates and behavioral actions) and the conflict resolution hierarchy.

There are five major steps in preparing to use the genetic programming paradigm, namely, determining:

(1) the set of terminals,
(2) the set of functions,
(3) the fitness function,
(4) the parameters and variables for controlling the run, and
(5) the criterion for designating a result and terminating a run.

The first major step in preparing to use genetic programming is to identify the set of terminals. The genetic programming paradigm genetically creates a computer program that takes certain inputs and produces outputs in order to successfully perform a specified task. The inputs to this program usually come from the statement of the problem. For the wall following problem, the potential inputs to the computer program consist of the 13 available sensors. These are the same 13 sensors which one would use if one were attempting to perform wall following with the conventional robotic approach or the subsumption architecture.

In reviewing the 13 sensors, we concluded that we had no use for the STOPPED sensor since our simulated robot could not be damaged by running into a wall in the course of a computer simulation. Moreover, we did not want our simulated robot to ever stop. Thus, we deleted the STOPPED sensor, the STOP primitive function, and the constant parameter for the danger zone DZ.

We retained Mataric's other two constant numerical parameters (i.e. the edging distance EDG and the minimum safe distance MSD). We retained Mataric's overall minimum sensor SS. However, we did not use any of her eight derived values representing specific subsets of sonar sensors. Human programmers find it convenient to create and refer to such intermediate variables in their programs.

Thus, our terminal set consisted of 15 items, namely,

```
T = {S00, S01, S02, S03, ...,
     S11, SS, MSD, EDG}
```

In other words, at each time step of the simulation, our simulated robot will have access to these 15 floating point values.

The second major step in preparing to use genetic programming is to identify a set of functions for the problem.

We start with the five given primitive motor functions that are part of the statement of this problem. As previously mentioned, we had no use for the STOP function. Since we want to evolve a subsumption architecture and we observed above that the subsumption architecture can be viewed as a composition of ordinary IF conditional functions, we included a single simple decision making function (IFLTE) in the function set. The function IFLTE (If-Less-Than- Or-Equal) takes four arguments. If the value of the first argument is less than or equal the value of the second argument, the third argument is evaluated and returned. Otherwise, the fourth argument is evaluated and returned.

We also included a connective function (PROGN2) in our function set. The connective function PROGN2 taking two arguments evaluates both of its arguments, in order, and returns the result of evaluating its second argument.

Thus, the function set F for this problem consists of four of the five given primitive motor functions (i.e. TR, TL, MF, and MB as described below), the decision function IFLTE, and the connective PROGN2. That is, the function set F is

```
F = {TR, TL, MF, MB, IFLTE,PROGN2}
```

The function TR (Turn Right) turns the robot 30 degrees to the right (i.e. clockwise).

The function TL (Turn Left) turns the robot 30 degrees to the left (i.e. counter-clockwise).

We achieved the same effect as the STOP function by letting our robot push up against the wall, and, if no change of state occurs after one time step, the robot is viewed as having stopped. Because we were not concerned with physically damaging our simulated robot, we did not include Mataric's primitive motor function STOP for stopping the robot (e.g. when it is about to invade the danger zone DZ and possibly damage itself).

The function MF (Move Forward) causes the robot to move 1.0 feet forward in the direction it is currently facing. If any of the six forward looking sonar sensors (i.e. S00 through S05) report a distance to any wall of less than 110% of the distance to be moved, no movement occurs.

The function MB (Move Backward) causes the robot to move 1.3 feet backwards. If any of the six backward looking sonar sensors (i.e. S06 though S11) report a distance to any wall of less than 110% of the distance to be moved, no movement occurs.

All sonar distances are dynamically recomputed after each execution of a move or turn. Each of the moving and turning functions returns the minimum of the two distances reported by the two sensors (i.e. S02 and S03) that look in the direction of forward movement (i.e. S02 representing the 11:30 o'clock direction and S03 representing the 12:30 o'clock direction).

The functions MF, MB, TR, and TL each take one time step (i.e. 1.0 seconds) to execute.

The third major step in preparing to use genetic programming is identification of the fitness function for evaluating how good a given computer program is at solving the problem at hand.

A wall following robot may be viewed as a robot that travels along the entire perimeter of the irregularly shaped room. Noting that the edging distance is 2.3 feet, we proceed to define the fitness measure for this problem by placing 2.3 foot square tiles along the perimeter of the room. Twelve such tiles fit along the 27.6 foot north wall and 12 such tiles fit along the 27.6 foot west wall. A total of 56 tiles are required to cover the entire periphery of the room.

Figure 5 shows the room with the 56 tiles (each with a filled circle at its center). The robot is shown in the middle of the room at its starting position (12, 16) facing in its starting direction (i.e. south).

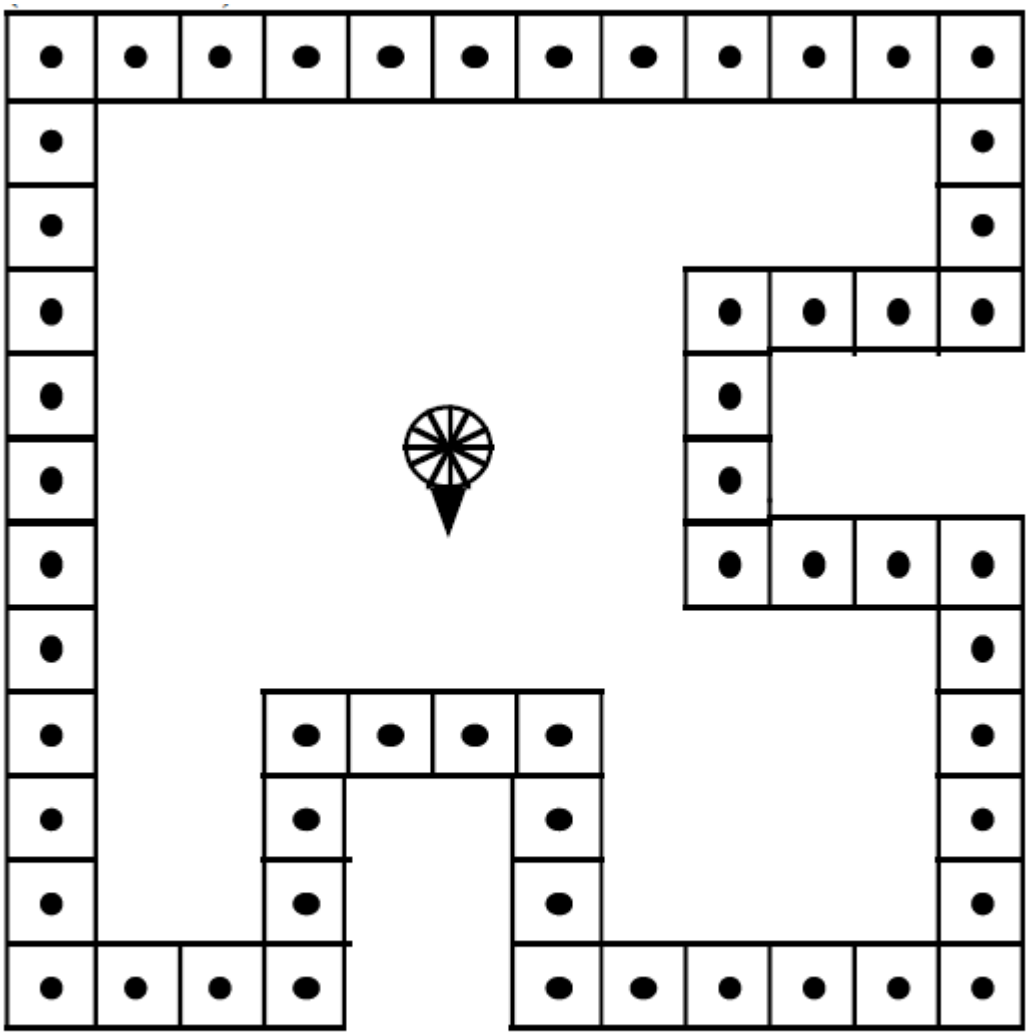


Figure 5: Room with 56 tiles on periphery showing robot at its starting position (12,16) facing south.

We defined the fitness of an individual S-expression in the population to be the number of tiles (from 0 to 56) that are touched by the robot within the allotted period of time (i.e. 400 time steps).

The fourth major step in preparing to use genetic programming is selecting the values of certain parameters. The population size is 1000 here. Each new generation is created from the preceding generation by applying the fitness proportionate reproduction operation to 10% of the population and by applying the crossover operation to 90% of the population (with both parents selected with a probability proportionate to fitness). In selecting crossover points, 90% were internal (function)

points of the tree and 10% were external (terminal) points of the tree. For the practical reason of conserving computer time, the depth of initial random S-expressions was limited to 4 and the depth of S-expressions created by crossover was limited to 15.

Finally, the fifth major step in preparing to use genetic programming is the selection of the criterion for terminating a run and accepting a result. We will terminate a given run when either (i) genetic programming produces a computer program which achieves the maximal value for fitness (i.e. 56 out of 56), or (ii) 101 generations have been run.

Note that in performing these five preparatory steps, we made use only of the information provided in the basic statement of the problem (with the modifications needed because we did not intend to allow our simulated robot ever to stop during the course of our computer simulations). We *did not* use any of Mataric's thoughtfully chosen subsets of sensors nor did we use any knowledge about the four task achieving behaviors which Mataric conceived and defined. We *did*, however, define a way to measure fitness in performing wall following. We *did* use the 12 sonar sensors, two of the three constant numerical parameters, and four of the five primitive motor functions that were part of the statement of the problem.

8 . RESULTS

In one run of the genetic programming paradigm on this problem, 57% of the individuals in the population in the initial random generation (i.e. generation 0) scored a fitness of zero (out of a possible 56). Many of these zeroscoring S-expressions merely caused the robot to turn without ever moving while others caused the robot to wander aimlessly in circles in the middle of the room. About 20% of the individuals from generation 0 were wall-bangers which scored precisely one because they headed for a wall and continued to push up against it.

The best single individual from generation 0 scored 17 (out of 56). This S-expression consists of 17 points (i.e. functions and terminals) and is shown below:

```
(IFLTE (PROGN2 MSD (TL))
       (IFLTE S06 S03 EDG (MF))
       (IFLTE MSD EDG S05 S06)
       (PROGN2 MSD (MF)))
```

Figure 6 shows the looping trajectory of the robot while executing this best-of-generation program for generation 0. The 39 filled circles along the periphery of the room represent the 39 of the 56 tiles that were not touched by the robot before it timed out. As can be seen, this individual starts in the middle of the room and circles on itself three times. It then begins a series of 11 loops which cause the robot to repeatedly hit the wall at irregular intervals. This looping leaves many intervening points along the wall untouched. This individual time outs on the west wall after 400 time steps. By generation 2, the best-of-generation individual scored 27. This S-expression consisted of 57 points.

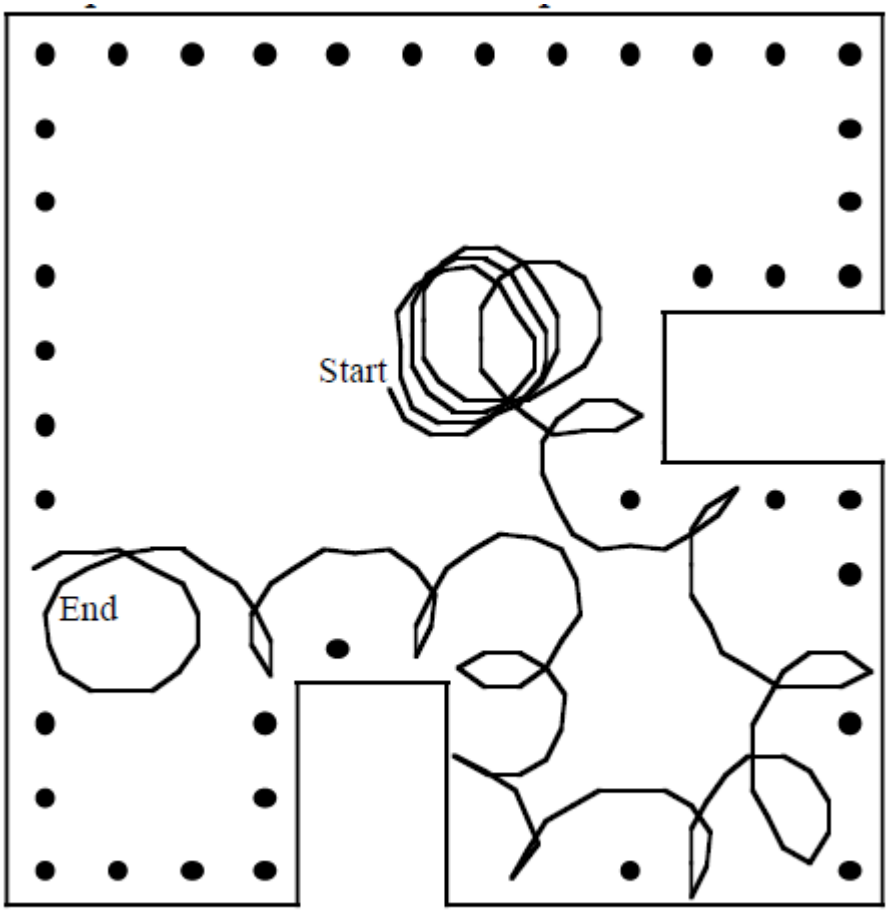


Figure 6: Looping trajectory from generation 0 of the best-of-generation individual (scoring 17).

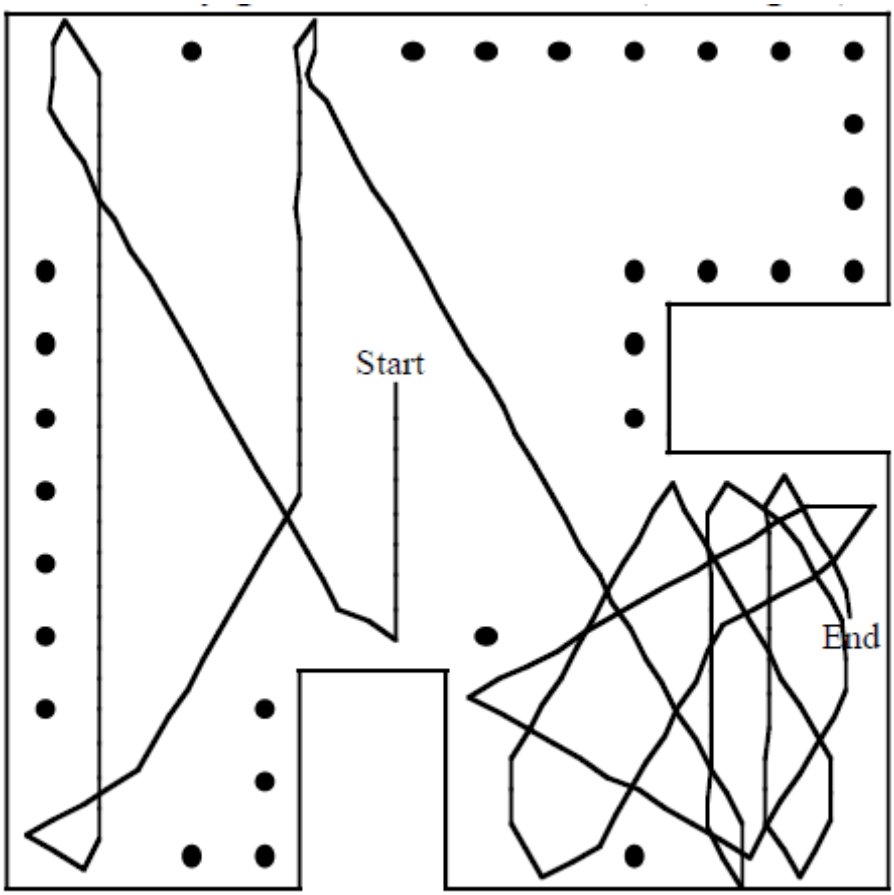


Figure 7: Ricocheting trajectory from generation 2 of the best-of-generation individual (scoring

Figure 7 shows the ricocheting trajectory of the robot while executing this best-of-generation program for generation 2. As can be seen, this individual causes the robot to touch occasional points on the periphery of the room as the robot ricochets around the room 16 times.

Although this ricocheting individual from generation 2 is far from perfect, it is considerably better than the static and aimless wandering individuals (both scoring zero) from generation 0, the wall-banging individuals (scoring one) from generation 0, and the best-of-generation looping individual from generation 0 (scoring 17).

By generation 14, the best-of-generation S-expression scored 49 and consisted of 45 points. Figure 8 shows the trajectory of the robot while executing this best-of-generation program for generation 14. After once reaching a wall, this individual slithers in broad snake like motions along the walls and never returns to the middle of the room. In scoring 49 out of 56, it misses five corners and two points in the middle of walls.

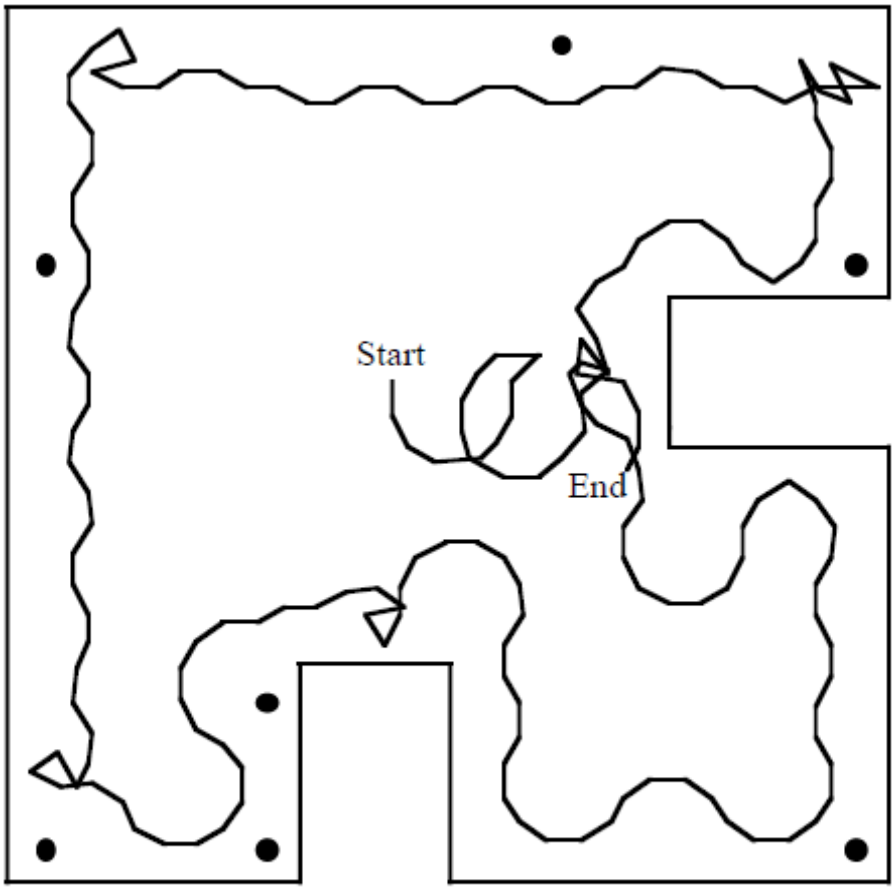


Figure 8: Broad snake like trajectory from generation 14 of best-of-generation individual (scoring 49 out of 56).

Finally, in generation 57, the best-of-generation S-expression scored a perfect 56 out of 56 This S-expression consisted of 145 points and is shown below:

```
(IFLTE (IFLTE S10 S05 S02 S05) (IFLTE (PROGN2 S11 S07) (PROGN2 (PROGN2 (PROGN2
S11 S05) (PROGN2 (PROGN2 S11 S05) (PROGN2 (MF) EDG))) SS) (PROGN2 (PROGN2
(IFLTE S02 (PROGN2 S11 S07) S04 (PROGN2 S11 S05)) (TL)) (MB)) (IFLTE S01 EDG
(TR) (TL))) (PROGN2 SS S08) (IFLTE (IFLTE (PROGN2 S11 S07) (PROGN2 (PROGN2
(PROGN2 S10 S05) (PROGN2 (PROGN2 S11 S05) (PROGN2 (MF) EDG))) SS) (PROGN2
(PROGN2 S01 (PROGN2 (IFLTE S07 (IFLTE S02 (PROGN2 (IFLTE SS EDG (TR) (TL))
(MB)) S04 S10) S04 S10) (TL))) (MB)) (IFLTE S01 EDG (TR) (TL))) (PROGN2 S05
SS) (PROGN2 (PROGN2 MSD (PROGN2 S11 S05)) (PROGN2 (IFLTE (PROGN2 (TR) (TR))
(PROGN2 S01 (PROGN2 (IFLTE S02 (TL) S04 (MB)) (TL))) (PROGN2 S07 (PROGN2
(PROGN2 (MF) EDG) EDG)) (IFLTE SS EDG (PROGN2 (PROGN2 (PROGN2 S02 S05) (PROGN2
(PROGN2 S11 S05) (PROGN2 (IFLTE S02 (TL) S04 S10) EDG))) SS) (TL))) S08))
(IFLTE SS EDG (TR) (TL))))
```

This program consists of a composition of conditional statements which test various sensors from the environment and invoke various given primitive motor functions of the robot in order to perform wall following. In other words, this program is a program in the subsumption architecture.

We can simplify this S-expression to the following S-expression containing 59 points:

```
(IFLTE (IFLTE S10 S05 S02 S05)
       (IFLTE S07 (PROGN2 (MF) SS)
              (PROGN2 (TL) (MB))
              (IFLTE S01 EDG (TR) (TL)))
       *
       (IFLTE (IFLTE S07 (PROGN2 (MF) SS)
              (PROGN (IFLTE SS EDG (TR) (TL))
                     (MB) (TL) (MB))
              (IFLTE S01 EDG (TR) (TL)))
       SS
       (IFLTE (PROGN2 (TR) (TR))
              (PROGN2 (IFLTE S02 (TL) * (MB))
                     (TL))
              (MF)
              (TL))
       (IFLTE SS EDG (TR) (TL))))
```

In this S-expression, the asterisks indicate subexpressions that are free of side-effects and which are just returned as the value of the expression (i.e. are executed, but are inconsequential).

Figure 9 shows the trajectory of the robot while executing this best-of-generation program for generation 57. This individual starts by briefly moving at random in the middle of the room. However, as soon as it reaches the wall, it moves along the wall and stays close to the wall. It touches l00% of the 56 tiles along the periphery of the room.

Note that the progressive change in size and shape of the individuals in the population is a characteristic of genetic programming. The size (i.e. 145 points) and particular hierarchical structure of the best-of-generation individual from generation 57 was not specified in advance. Instead, the entire structure evolved as result of reproduction, crossover, and the relentless pressure of fitness. That is, fitness caused the development of the structure.

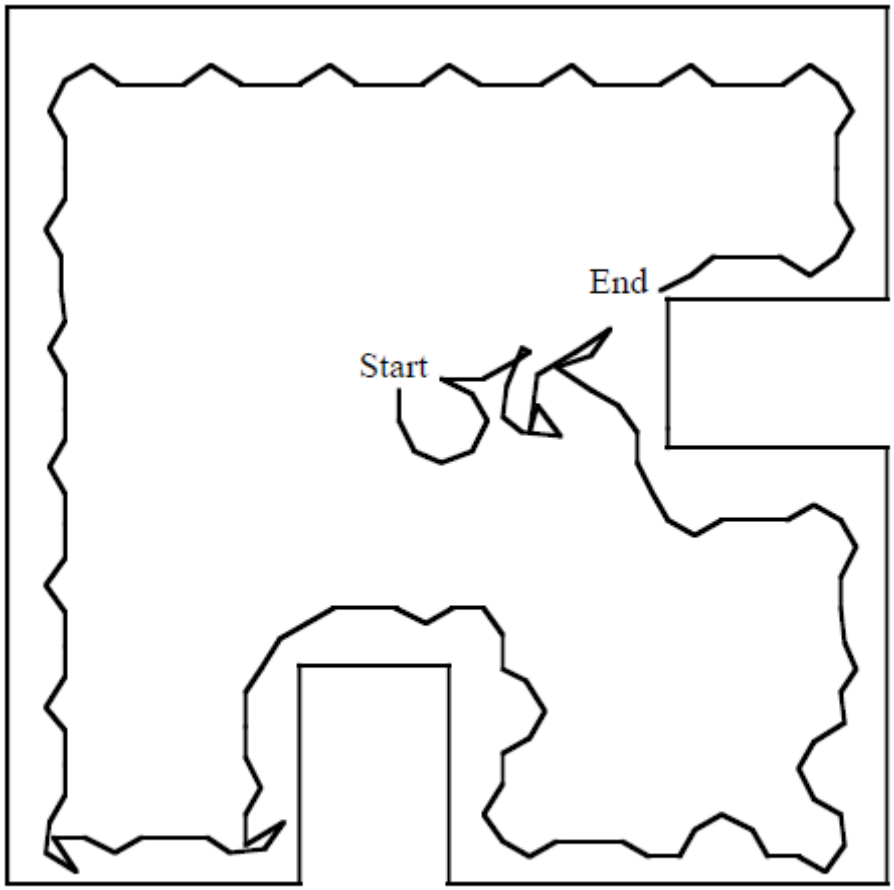


Figure 9: Wall following trajectory of the best-of-generation individual (scoring 56 out of 56) from generation 57.

Although a program written by a human programmer cannot be directly compared to the program generated using genetic programming, it is, nonetheless, interesting to note that the 145 points of this S-expression is similar to the 151 points in Mataric's four LISP programs.

We have obtained similar results on other runs of this problem.

9 . CONCLUSIONS

We demonstrated that it is possible to use the genetic programming paradigm to breed a computer program to enable a robot to follow the wall of an irregular room.

The program we discovered consisted of a composition of conditional statements which tested various sensors from the environment and invoked various given primitive motor functions of the robot in order to perform wall following. In other words, this program is a program in the subsumption architecture. Thus, we have demonstrated the

evolution of a program in the subsumption architecture using an evolutionary process that evolves structures guided only by a fitness measure.

The fact that it is possible to evolve a subsumption

1 0 . ACKNOWLEDGMENTS

James P. Rice of the Knowledge Systems Laboratory at Stanford University made numerous contributions in connection with the computer programming of the above.

1 1 . REFERENCES

- Belew, Richard and Booker, Lashon (editors) *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, Ca: Morgan Kaufmann Publishers Inc. 1991.
- Brooks, Rodney. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*. 2(1) March 1986.
- Brooks, Rodney. A robot that walks: emergent behaviors from a carefully evolved network. *Neural Computation* 1(2), 253-262. 1989.
- Davis, Lawrence (editor) Genetic Algorithms and Simulated Annealing London: Pittman l987.
- Davis, Lawrence. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.1991.
- Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley l989.
- Holland, John H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press 1975.
- Koza, John R. Hierarchical genetic algorithms operating on populations of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann 1989.
- Koza, John R. *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Stanford University Computer Science Dept. Technical Report STAN-CS-90-1314. June 1990.
- Koza, John R. Genetic evolution and coevolution of computer programs. In Langton, Christopher, Taylor, Charles, Farmer, J. Doyne, and Rasmussen, Steen (editors). *Artificial Life II, SFI Studies in the Sciences of Complexity*. Volume X. Redwood City, CA: Addison-Wesley 1991. 603-629. 1991a.
- Koza, John R. Evolution and co-evolution of computer programs to control independentacting agents. In Meyer and Wilson below. 1991b.
- Koza, John R. Evolving a computer program to generate random numbers using the genetic programming paradigm. In Belew and Booker above. 1991c.
- Koza, John R. A hierarchical approach to learning the Boolean multiplexer function. In Rawlins below. 1991d.

- Koza, John R. *Genetic Programming*. Cambridge, MA: MIT Press, 1992 (forthcoming). Koza, John R. and Keane, Martin A. Genetic breeding of non-linear optimal control strategies for broom balancing. In *Proceedings of the Ninth International Conference on Analysis and Optimization of Systems.* Berlin: Springer-Verlag, 1990a.
- Koza, John R. and Keane, Martin. Cart centering and broom balancing by genetically breeding populations of control strategy programs. In *Proceedings of International Joint Conference on Neural Networks, Washington, January, 1990*. Volume I. Hillsdale, NJ: Lawrence Erlbaum 1990b.
- Koza, John R. and Rice, James P. Genetic generation of both the weights and architecture for a neural network. In *Proceedings of International Joint Conference on Neural Networks, Seattle, July 1991*. 1991a
- Koza, John R. and Rice, James P. A genetic approach to artificial intelligence. In C. G. Langton (editor) *Artificial Life II VideoProceedings*. Addison-Wesley 1991. 1991b.
- Meyer, Jean-Arcady and Wilson, Stewart W. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. Paris. September 24-28, 1990. MIT Press, Cambridge, MA, 1991.
- Mataric, Maja J. *A Distributed Model for Mobile Robot Environment-Learning and Navigation*. MIT Artificial Intelligence Laboratory technical report AI-TR-1228. May 1990.
- Langton, Christopher G. *Artificial Life, Santa Fe Institute Studies in the Sciences of Complexity*. Volume VI. Redwood City, CA: Addison-Wesley. 1989.
- Rawlins, Gregory (editor). *Proceedings of Workshop on the Foundations of Genetic Algorithms and Classifier Systems. Bloomington, Indiana. July 15-18, 1990*. San Mateo, CA: Morgan Kaufmann 1991.
- Steels, Luc. Towards a theory of emergent functionality. In Meyer, Jean-Arcady and Wilson, Stewart W. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. Paris. September 24-28, 1990. Cambridge, MA: MIT Press 1991.

Anexo 2: Tablas de datos

- Resultados del experimento 1:

Repetición	Raw Fitness	Adj. Fitness	Best nodes rpb 0	Best depth rpb 0	Generations	Time (s)
exp_0.0	490.000	0.0200	87	13	50	1029
exp_0.1	600.000	0.0164	71	11	50	838
exp_0.2	580.000	0.0169	69	8	50	977
exp_0.3	580.000	0.0169	67	9	50	877
exp_0.4	440.000	0.0222	87	9	50	953
exp_0.5	510.000	0.0192	69	12	50	967
exp_0.6	460.000	0.0213	99	16	50	1001
exp_0.7	480.000	0.0204	87	9	50	1088
exp_0.8	460.000	0.0213	99	10	50	1063
exp_0.9	600.000	0.0164	91	9	50	856
exp_1.0	570.000	0.0172	75	7	50	520
exp_1.1	620.000	0.0159	51	8	50	503
exp_1.2	650.000	0.0152	99	8	50	544
exp_1.3	640.000	0.0154	73	8	50	522
exp_1.4	620.000	0.0159	59	6	50	496
exp_1.5	670.000	0.0147	79	8	50	436
exp_1.6	640.000	0.0154	69	7	50	512
exp_1.7	620.000	0.0159	57	8	50	608
exp_1.8	640.000	0.0154	73	8	50	525
exp_1.9	620.000	0.0159	43	8	50	504
exp_2.0	450.000	0.0217	91	9	100	1851
exp_2.1	510.000	0.0192	89	8	100	1721
exp_2.2	490.000	0.0200	97	8	100	1660
exp_2.3	590.000	0.0167	91	9	100	1387
exp_2.4	560.000	0.0175	85	13	100	1478
exp_2.5	620.000	0.0159	83	11	100	1324
exp_2.6	610.000	0.0161	87	9	100	1254
exp_2.7	580.000	0.0169	99	12	100	1529
exp_2.8	520.000	0.0189	85	10	100	1544
exp_2.9	450.000	0.0217	67	10	100	1158
exp_3.0	580.000	0.0169	89	9	100	862
exp_3.1	650.000	0.0152	97	8	100	694
exp_3.2	660.000	0.0149	95	9	100	1043
exp_3.3	660.000	0.0149	93	9	100	809

exp_3.4	600.000	0.0164	75	8	100	792
exp_3.5	630.000	0.0156	89	9	100	719
exp_3.6	630.000	0.0156	97	9	100	833
exp_3.7	630.000	0.0156	61	9	100	962
exp_3.8	610.000	0.0161	75	9	100	1168
exp_3.9	620.000	0.0159	89	7	100	829
exp_4.0	550.000	0.0179	89	10	150	2003
exp_4.1	510.000	0.0192	71	10	150	1843
exp_4.2	530.000	0.0185	85	11	150	1608
exp_4.3	530.000	0.0185	73	11	150	1799
exp_4.4	620.000	0.0159	71	9	150	1573
exp_4.5	620.000	0.0159	75	8	150	1723
exp_4.6	460.000	0.0213	89	11	150	1820
exp_4.7	570.000	0.0172	87	12	150	1943
exp_4.8	540.000	0.0182	95	10	150	1815
exp_4.9	530.000	0.0185	93	10	150	2123
exp_5.0	660.000	0.0149	73	9	150	1132
exp_5.1	650.000	0.0152	83	8	150	1184
exp_5.2	620.000	0.0159	93	8	150	1349
exp_5.3	650.000	0.0152	99	9	150	975
exp_5.4	650.000	0.0152	77	8	150	990
exp_5.5	680.000	0.0145	95	8	150	1445
exp_5.6	640.000	0.0154	83	9	150	1029
exp_5.7	570.000	0.0172	81	7	150	1546
exp_5.8	640.000	0.0154	69	9	150	1103
exp_5.9	660.000	0.0149	91	8	150	1488
exp_6.0	540.000	0.0182	63	8	200	2080
exp_6.1	600.000	0.0164	77	13	200	1788
exp_6.2	590.000	0.0167	87	12	200	2466
exp_6.3	600.000	0.0164	99	12	200	2630
exp_6.4	620.000	0.0159	99	11	200	2218
exp_6.5	550.000	0.0179	89	12	200	1977
exp_6.6	600.000	0.0164	93	9	200	2381
exp_6.7	490.000	0.0200	91	9	200	2046
exp_6.8	570.000	0.0172	85	9	200	2185
exp_6.9	600.000	0.0164	93	13	200	2388
exp_7.0	620.000	0.0159	93	8	200	1830
exp_7.1	620.000	0.0159	97	10	200	1025
exp_7.2	630.000	0.0156	69	7	200	1087
exp_7.3	630.000	0.0156	81	7	200	1701
exp_7.4	640.000	0.0154	41	8	200	1318
exp_7.5	630.000	0.0156	83	9	200	1586

exp_7.6	640.000	0.0154	89	9	200	1292
exp_7.7	610.000	0.0161	73	8	200	2220
exp_7.8	650.000	0.0152	95	10	200	1629
exp_7.9	640.000	0.0154	41	8	200	1525
exp_8.0	460.000	0.0213	85	12	50	8065
exp_8.1	460.000	0.0213	87	20	50	7840
exp_8.2	470.000	0.0208	99	17	50	8082
exp_8.3	440.000	0.0222	97	12	50	8224
exp_8.4	430.000	0.0227	91	25	50	8547
exp_8.5	420.000	0.0233	99	17	50	8230
exp_8.6	410.000	0.0238	95	12	50	7918
exp_8.7	500.000	0.0196	79	11	50	5201
exp_8.8	460.000	0.0213	85	16	50	5081
exp_8.9	430.000	0.0227	97	14	50	5154
exp_9.0	610.000	0.0161	91	8	50	871
exp_9.1	600.000	0.0164	77	8	50	847
exp_9.2	600.000	0.0164	51	8	50	850
exp_9.3	680.000	0.0145	87	9	50	795
exp_9.4	650.000	0.0152	83	9	50	848
exp_9.5	640.000	0.0154	63	7	50	784
exp_9.6	620.000	0.0159	97	9	50	812
exp_9.7	630.000	0.0156	47	6	50	712
exp_9.8	600.000	0.0164	49	8	50	708
exp_9.9	640.000	0.0154	71	8	50	589
exp_10.0	460.000	0.0213	93	12	100	5778
exp_10.1	500.000	0.0196	75	17	100	5537
exp_10.2	410.000	0.0238	91	16	100	5660
exp_10.3	480.000	0.0204	93	16	100	6071
exp_10.4	490.000	0.0200	79	15	100	5784
exp_10.5	440.000	0.0222	97	12	100	6136
exp_10.6	540.000	0.0182	87	23	100	5840
exp_10.7	350.000	0.0278	87	14	100	6023
exp_10.8	470.000	0.0208	99	12	100	5743
exp_10.9	510.000	0.0192	97	9	100	6171
exp_11.0	680.000	0.0145	61	6	100	1030
exp_11.1	630.000	0.0156	75	9	100	1565
exp_11.2	650.000	0.0152	51	7	100	1433
exp_11.3	670.000	0.0147	79	8	100	1564
exp_11.4	610.000	0.0161	51	8	100	1390
exp_11.5	630.000	0.0156	53	7	100	1318
exp_11.6	510.000	0.0192	69	8	100	1305
exp_11.7	610.000	0.0161	71	9	100	1303

exp_11.8	650.000	0.0152	65	7	100	794
exp_11.9	590.000	0.0167	69	10	100	989
exp_12.0	430.000	0.0227	93	13	150	6586
exp_12.1	460.000	0.0213	97	14	150	6821
exp_12.2	470.000	0.0208	93	14	150	6795
exp_12.3	420.000	0.0233	99	25	150	7150
exp_12.4	470.000	0.0208	93	15	150	6668
exp_12.5	470.000	0.0208	97	18	150	5928
exp_12.6	470.000	0.0208	95	10	150	4405
exp_12.7	480.000	0.0204	93	20	150	4346
exp_12.8	460.000	0.0213	89	12	150	4054
exp_12.9	460.000	0.0213	93	16	150	4247
exp_13.0	660.000	0.0149	77	8	150	1018
exp_13.1	620.000	0.0159	83	9	150	794
exp_13.2	640.000	0.0154	85	9	150	984
exp_13.3	670.000	0.0147	87	9	150	1146
exp_13.4	580.000	0.0169	63	9	150	1147
exp_13.5	590.000	0.0167	91	9	150	1224
exp_13.6	660.000	0.0149	87	9	150	1022
exp_13.7	630.000	0.0156	51	8	150	973
exp_13.8	640.000	0.0154	69	9	150	1492
exp_13.9	650.000	0.0152	39	6	150	1071
exp_14.0	480.000	0.0204	87	12	200	4263
exp_14.1	480.000	0.0204	89	13	200	4624
exp_14.2	450.000	0.0217	99	11	200	4760
exp_14.3	440.000	0.0222	97	15	200	4896
exp_14.4	440.000	0.0222	89	14	200	4950
exp_14.5	410.000	0.0238	91	16	200	4541
exp_14.6	510.000	0.0192	93	12	200	4961
exp_14.7	370.000	0.0263	95	14	200	5201
exp_14.8	460.000	0.0213	97	21	200	4742
exp_14.9	380.000	0.0256	99	15	200	4684
exp_15.0	640.000	0.0154	77	8	200	1587
exp_15.1	640.000	0.0154	77	8	200	1402
exp_15.2	620.000	0.0159	61	10	200	1361
exp_15.3	640.000	0.0154	57	7	200	1510
exp_15.4	640.000	0.0154	43	8	200	1436
exp_15.5	640.000	0.0154	55	9	200	776
exp_15.6	690.000	0.0143	67	8	200	1652
exp_15.7	630.000	0.0156	51	7	200	1229
exp_15.8	660.000	0.0149	61	7	200	1829
exp_15.9	520.000	0.0189	83	8	200	1579

exp_16.0	510.000	0.0192	91	12	50	5439
exp_16.1	380.000	0.0256	97	19	50	6137
exp_16.2	430.000	0.0227	97	14	50	6005
exp_16.3	430.000	0.0227	99	15	50	5439
exp_16.4	450.000	0.0217	93	15	50	5808
exp_16.5	500.000	0.0196	89	14	50	5229
exp_16.6	430.000	0.0227	95	16	50	5872
exp_16.7	420.000	0.0233	93	14	50	5566
exp_16.8	450.000	0.0217	93	13	50	5592
exp_16.9	470.000	0.0208	81	10	50	5287
exp_17.0	700.000	0.0141	75	8	50	629
exp_17.1	640.000	0.0154	55	7	50	628
exp_17.2	600.000	0.0164	55	8	50	531
exp_17.3	580.000	0.0169	65	8	50	633
exp_17.4	660.000	0.0149	93	9	50	715
exp_17.5	630.000	0.0156	57	8	50	529
exp_17.6	630.000	0.0156	77	8	50	611
exp_17.7	650.000	0.0152	63	8	50	640
exp_17.8	680.000	0.0145	81	9	50	611
exp_17.9	630.000	0.0156	93	10	50	667
exp_18.0	490.000	0.0200	87	15	100	6402
exp_18.1	410.000	0.0238	99	19	100	6580
exp_18.2	460.000	0.0213	97	13	100	6145
exp_18.3	440.000	0.0222	93	20	100	5365
exp_18.4	380.000	0.0256	99	20	100	5542
exp_18.5	380.000	0.0256	99	13	100	5635
exp_18.6	450.000	0.0217	93	12	100	5531
exp_18.7	370.000	0.0263	99	13	100	5579
exp_18.8	490.000	0.0200	95	12	100	5657
exp_18.9	440.000	0.0222	95	20	100	5400
exp_19.0	640.000	0.0154	43	6	100	591
exp_19.1	620.000	0.0159	75	9	100	901
exp_19.2	640.000	0.0154	43	7	100	803
exp_19.3	610.000	0.0161	59	9	100	744
exp_19.4	640.000	0.0154	99	9	100	585
exp_19.5	660.000	0.0149	67	9	100	835
exp_19.6	650.000	0.0152	91	9	100	774
exp_19.7	650.000	0.0152	75	9	100	1083
exp_19.8	650.000	0.0152	45	6	100	712
exp_19.9	530.000	0.0185	53	7	100	1021
exp_20.0	480.000	0.0204	81	13	150	5319
exp_20.1	510.000	0.0192	93	13	150	5942

exp_20.2	400.000	0.0244	97	10	150	5521
exp_20.3	320.000	0.0303	99	17	150	6114
exp_20.4	410.000	0.0238	89	14	150	5627
exp_20.5	420.000	0.0233	87	12	150	5475
exp_20.6	440.000	0.0222	89	18	150	6246
exp_20.7	420.000	0.0233	99	17	150	6449
exp_20.8	440.000	0.0222	99	13	150	6879
exp_20.9	470.000	0.0208	99	14	150	6644
exp_21.0	560.000	0.0175	87	8	150	1264
exp_21.1	560.000	0.0175	69	7	150	1316
exp_21.2	630.000	0.0156	91	8	150	1162
exp_21.3	650.000	0.0152	43	8	150	787
exp_21.4	540.000	0.0182	75	10	150	1352
exp_21.5	690.000	0.0143	97	9	150	1241
exp_21.6	620.000	0.0159	65	8	150	1298
exp_21.7	440.000	0.0222	95	8	150	982
exp_21.8	610.000	0.0161	71	9	150	1170
exp_21.9	660.000	0.0149	65	9	150	872
exp_22.0	480.000	0.0204	95	15	200	7222
exp_22.1	470.000	0.0208	93	11	200	6528
exp_22.2	460.000	0.0213	91	12	200	7124
exp_22.3	460.000	0.0213	79	10	200	6423
exp_22.4	410.000	0.0238	93	11	200	6276
exp_22.5	450.000	0.0217	97	16	200	6874
exp_22.6	440.000	0.0222	87	16	200	6584
exp_22.7	410.000	0.0238	85	17	200	6228
exp_22.8	510.000	0.0192	89	17	200	6447
exp_22.9	410.000	0.0238	93	11	200	6647
exp_23.0	650.000	0.0152	85	9	200	1690
exp_23.1	620.000	0.0159	47	7	200	1054
exp_23.2	630.000	0.0156	35	6	200	1041
exp_23.3	650.000	0.0152	9	4	200	1759
exp_23.4	630.000	0.0156	79	9	200	1106
exp_23.5	640.000	0.0154	73	8	200	1373
exp_23.6	630.000	0.0156	29	7	200	1066
exp_23.7	640.000	0.0154	13	4	200	1497
exp_23.8	640.000	0.0154	59	8	200	789
exp_23.9	610.000	0.0161	85	8	200	1599
exp_24.0	450.000	0.0217	93	13	50	6894
exp_24.1	450.000	0.0217	97	10	50	6880
exp_24.2	370.000	0.0263	95	19	50	7251
exp_24.3	470.000	0.0208	95	15	50	7264

exp_24.4	460.000	0.0213	99	11	50	7188
exp_24.5	380.000	0.0256	93	13	50	7059
exp_24.6	370.000	0.0263	95	13	50	7026
exp_24.7	440.000	0.0222	87	12	50	7557
exp_24.8	450.000	0.0217	97	12	50	11383
exp_24.9	470.000	0.0208	91	17	50	10807
exp_25.0	630.000	0.0156	77	8	50	665
exp_25.1	670.000	0.0147	79	9	50	523
exp_25.2	620.000	0.0159	67	10	50	489
exp_25.3	620.000	0.0159	57	9	50	694
exp_25.4	620.000	0.0159	47	8	50	488
exp_25.5	660.000	0.0149	67	7	50	639
exp_25.6	630.000	0.0156	63	9	50	542
exp_25.7	590.000	0.0167	51	7	50	625
exp_25.8	600.000	0.0164	37	6	50	559
exp_25.9	620.000	0.0159	49	7	50	493
exp_26.0	500.000	0.0196	91	17	100	8195
exp_26.1	460.000	0.0213	85	15	100	8753
exp_26.2	460.000	0.0213	93	17	100	8276
exp_26.3	460.000	0.0213	97	15	100	8256
exp_26.4	470.000	0.0208	93	9	100	7695
exp_26.5	470.000	0.0208	95	11	100	8065
exp_26.6	480.000	0.0204	87	19	100	7793
exp_26.7	470.000	0.0208	99	16	100	7383
exp_26.8	360.000	0.0270	97	13	100	7163
exp_26.9	490.000	0.0200	97	14	100	6723
exp_27.0	700.000	0.0141	75	8	100	751
exp_27.1	660.000	0.0149	97	9	100	689
exp_27.2	630.000	0.0156	83	8	100	1068
exp_27.3	650.000	0.0152	95	9	100	748
exp_27.4	640.000	0.0154	85	9	100	689
exp_27.5	570.000	0.0172	69	8	100	810
exp_27.6	630.000	0.0156	63	7	100	770
exp_27.7	660.000	0.0149	83	9	100	687
exp_27.8	620.000	0.0159	77	7	100	1020
exp_27.9	650.000	0.0152	77	9	100	620
exp_28.0	490.000	0.0200	71	13	150	7250
exp_28.1	430.000	0.0227	93	16	150	7298
exp_28.2	450.000	0.0217	95	16	150	7237
exp_28.3	480.000	0.0204	85	10	150	7252
exp_28.4	490.000	0.0200	89	11	150	7478
exp_28.5	470.000	0.0208	85	12	150	7711

exp_28.6	450.000	0.0217	87	15	150	7900
exp_28.7	470.000	0.0208	97	13	150	8111
exp_28.8	430.000	0.0227	93	17	150	8590
exp_28.9	480.000	0.0204	91	17	150	7828
exp_29.0	600.000	0.0164	65	9	150	1205
exp_29.1	640.000	0.0154	79	9	150	1226
exp_29.2	650.000	0.0152	57	7	150	1290
exp_29.3	650.000	0.0152	57	7	150	1208
exp_29.4	590.000	0.0167	47	7	150	1065
exp_29.5	580.000	0.0169	79	9	150	1145
exp_29.6	620.000	0.0159	77	7	150	1209
exp_29.7	600.000	0.0164	99	8	150	1094
exp_29.8	590.000	0.0167	89	8	150	1883
exp_29.9	670.000	0.0147	97	9	150	1491
exp_30.0	480.000	0.0204	93	12	200	8685
exp_30.1	450.000	0.0217	95	17	200	8823
exp_30.2	420.000	0.0233	83	10	200	7808
exp_30.3	430.000	0.0227	95	16	200	8536
exp_30.4	410.000	0.0238	99	11	200	8222
exp_30.5	400.000	0.0244	91	20	200	8616
exp_30.6	480.000	0.0204	85	15	200	8395
exp_30.7	500.000	0.0196	87	13	200	8446
exp_30.8	450.000	0.0217	95	11	200	8392
exp_30.9	500.000	0.0196	69	13	200	8087
exp_31.0	600.000	0.0164	77	7	200	1807
exp_31.1	590.000	0.0167	79	8	200	1900
exp_31.2	670.000	0.0147	51	6	200	1500
exp_31.3	650.000	0.0152	53	8	200	1083
exp_31.4	650.000	0.0152	71	9	200	897
exp_31.5	590.000	0.0167	65	9	200	1609
exp_31.6	640.000	0.0154	57	8	200	746
exp_31.7	600.000	0.0164	81	9	200	1327
exp_31.8	600.000	0.0164	79	9	200	1693
exp_31.9	620.000	0.0159	87	8	200	1965

- Resultados del experimento 2:

Repetición	Raw Fitness	Adj. Fitness	Best nodes rpb 0	Best depth rpb 0	Generations	Time (s)
exp_0.0	590.000	0.0167	35	8	50	804
exp_0.1	550.000	0.0179	69	10	50	894
exp_0.2	540.000	0.0182	91	14	50	1001
exp_0.3	570.000	0.0172	83	10	50	790
exp_0.4	580.000	0.0169	95	11	50	964
exp_0.5	490.000	0.0200	89	12	50	1038
exp_0.6	540.000	0.0182	99	10	50	1050
exp_0.7	570.000	0.0172	97	11	50	1216
exp_0.8	590.000	0.0167	77	11	50	1145
exp_0.9	570.000	0.0172	75	12	50	964
exp_1.0	650.000	0.0152	35	5	50	489
exp_1.1	640.000	0.0154	71	7	50	563
exp_1.2	640.000	0.0154	57	10	50	454
exp_1.3	630.000	0.0156	67	9	50	489
exp_1.4	680.000	0.0145	67	8	50	537
exp_1.5	600.000	0.0164	69	8	50	504
exp_1.6	620.000	0.0159	79	9	50	622
exp_1.7	650.000	0.0152	99	9	50	629
exp_1.8	700.000	0.0141	47	7	50	546
exp_1.9	650.000	0.0152	75	9	50	483
exp_2.0	540.000	0.0182	85	11	100	1189
exp_2.1	560.000	0.0175	99	11	100	1543
exp_2.2	480.000	0.0204	85	12	100	1336
exp_2.3	560.000	0.0175	89	10	100	1523
exp_2.4	310.000	0.0313	93	10	100	1386
exp_2.5	530.000	0.0185	81	11	100	1343
exp_2.6	590.000	0.0167	95	15	100	1226
exp_2.7	590.000	0.0167	99	9	100	1680
exp_2.8	550.000	0.0179	77	9	100	1227
exp_2.9	440.000	0.0222	87	9	100	1407
exp_3.0	630.000	0.0156	87	8	100	827
exp_3.1	610.000	0.0161	71	9	100	912
exp_3.2	580.000	0.0169	63	9	100	827
exp_3.3	590.000	0.0167	99	8	100	882
exp_3.4	630.000	0.0156	91	9	100	867
exp_3.5	680.000	0.0145	51	6	100	685
exp_3.6	620.000	0.0159	81	8	100	955

exp_3.7	630.000	0.0156	91	8	100	759
exp_3.8	620.000	0.0159	99	9	100	1069
exp_3.9	550.000	0.0179	35	8	100	837
exp_4.0	530.000	0.0185	69	9	150	1730
exp_4.1	580.000	0.0169	53	13	150	1512
exp_4.2	590.000	0.0167	99	12	150	1544
exp_4.3	600.000	0.0164	69	8	150	1646
exp_4.4	490.000	0.0200	97	11	150	2387
exp_4.5	520.000	0.0189	85	9	150	1863
exp_4.6	620.000	0.0159	65	8	150	1724
exp_4.7	580.000	0.0169	85	10	150	1855
exp_4.8	550.000	0.0179	89	11	150	1901
exp_4.9	510.000	0.0192	89	10	150	2268
exp_5.0	600.000	0.0164	85	9	150	610
exp_5.1	680.000	0.0145	99	9	150	989
exp_5.2	660.000	0.0149	77	8	150	1172
exp_5.3	650.000	0.0152	57	7	150	748
exp_5.4	620.000	0.0159	55	8	150	1237
exp_5.5	620.000	0.0159	75	8	150	1177
exp_5.6	590.000	0.0167	39	6	150	1116
exp_5.7	650.000	0.0152	59	8	150	786
exp_5.8	620.000	0.0159	91	8	150	1690
exp_5.9	700.000	0.0141	87	8	150	1661
exp_6.0	550.000	0.0179	75	8	200	1879
exp_6.1	570.000	0.0172	91	10	200	2384
exp_6.2	480.000	0.0204	97	13	200	2396
exp_6.3	340.000	0.0286	81	11	200	2393
exp_6.4	520.000	0.0189	89	12	200	2290
exp_6.5	510.000	0.0192	87	15	200	2136
exp_6.6	500.000	0.0196	93	12	200	1915
exp_6.7	590.000	0.0167	79	9	200	1863
exp_6.8	600.000	0.0164	71	9	200	1801
exp_6.9	580.000	0.0169	99	11	200	1865
exp_7.0	650.000	0.0152	65	7	200	1644
exp_7.1	630.000	0.0156	63	8	200	1405
exp_7.2	630.000	0.0156	91	9	200	1468
exp_7.3	650.000	0.0152	49	9	200	979
exp_7.4	650.000	0.0152	51	7	200	999
exp_7.5	650.000	0.0152	89	10	200	1116
exp_7.6	610.000	0.0161	79	8	200	1344
exp_7.7	660.000	0.0149	71	8	200	1234
exp_7.8	610.000	0.0161	73	8	200	1511

exp_7.9	520.000	0.0189	95	8	200	1903
exp_8.0	420.000	0.0233	89	12	50	2575
exp_8.1	570.000	0.0172	65	12	50	2679
exp_8.2	460.000	0.0213	93	16	50	2543
exp_8.3	490.000	0.0200	95	18	50	2637
exp_8.4	480.000	0.0204	93	13	50	2561
exp_8.5	460.000	0.0213	89	14	50	2442
exp_8.6	450.000	0.0217	89	11	50	2447
exp_8.7	460.000	0.0213	93	13	50	2569
exp_8.8	410.000	0.0238	85	13	50	2567
exp_8.9	470.000	0.0208	87	11	50	2985
exp_9.0	630.000	0.0156	29	6	50	422
exp_9.1	630.000	0.0156	85	9	50	532
exp_9.2	700.000	0.0141	97	8	50	521
exp_9.3	680.000	0.0145	97	8	50	597
exp_9.4	640.000	0.0154	89	7	50	483
exp_9.5	570.000	0.0172	93	9	50	650
exp_9.6	640.000	0.0154	81	8	50	521
exp_9.7	560.000	0.0175	91	9	50	570
exp_9.8	630.000	0.0156	65	8	50	672
exp_9.9	660.000	0.0149	69	8	50	620
exp_10.0	430.000	0.0227	81	12	100	3292
exp_10.1	440.000	0.0222	91	15	100	3437
exp_10.2	390.000	0.0250	97	16	100	3418
exp_10.3	430.000	0.0227	95	15	100	3240
exp_10.4	480.000	0.0204	81	11	100	3235
exp_10.5	470.000	0.0208	93	26	100	3505
exp_10.6	430.000	0.0227	85	16	100	3387
exp_10.7	430.000	0.0227	91	21	100	3396
exp_10.8	440.000	0.0222	95	10	100	3243
exp_10.9	470.000	0.0208	91	17	100	3181
exp_11.0	630.000	0.0156	73	8	100	768
exp_11.1	620.000	0.0159	83	9	100	659
exp_11.2	630.000	0.0156	43	7	100	1020
exp_11.3	650.000	0.0152	83	9	100	1015
exp_11.4	640.000	0.0154	87	9	100	866
exp_11.5	640.000	0.0154	37	7	100	655
exp_11.6	660.000	0.0149	61	8	100	558
exp_11.7	640.000	0.0154	99	8	100	762
exp_11.8	650.000	0.0152	99	9	100	792
exp_11.9	690.000	0.0143	81	9	100	1253
exp_12.0	410.000	0.0238	97	14	150	3970

exp_12.1	500.000	0.0196	77	12	150	3764
exp_12.2	420.000	0.0233	89	10	150	3805
exp_12.3	430.000	0.0227	85	12	150	3897
exp_12.4	450.000	0.0217	95	15	150	3512
exp_12.5	370.000	0.0263	99	15	150	4112
exp_12.6	390.000	0.0250	91	14	150	3678
exp_12.7	460.000	0.0213	95	12	150	3531
exp_12.8	430.000	0.0227	91	12	150	3492
exp_12.9	490.000	0.0200	93	12	150	3038
exp_13.0	650.000	0.0152	91	9	150	1313
exp_13.1	640.000	0.0154	69	9	150	994
exp_13.2	540.000	0.0182	99	8	150	1147
exp_13.3	550.000	0.0179	55	9	150	685
exp_13.4	620.000	0.0159	47	8	150	965
exp_13.5	650.000	0.0152	49	9	150	836
exp_13.6	610.000	0.0161	81	8	150	1185
exp_13.7	600.000	0.0164	93	10	150	868
exp_13.8	650.000	0.0152	83	8	150	984
exp_13.9	640.000	0.0154	53	9	150	1216
exp_14.0	440.000	0.0222	95	16	200	3735
exp_14.1	460.000	0.0213	95	13	200	3703
exp_14.2	430.000	0.0227	95	12	200	4352
exp_14.3	450.000	0.0217	89	13	200	4591
exp_14.4	450.000	0.0217	97	17	200	4634
exp_14.5	510.000	0.0192	83	14	200	4451
exp_14.6	460.000	0.0213	95	15	200	3708
exp_14.7	430.000	0.0227	95	14	200	3777
exp_14.8	450.000	0.0217	99	18	200	3981
exp_14.9	520.000	0.0189	91	12	200	4055
exp_15.0	670.000	0.0147	65	9	200	2003
exp_15.1	660.000	0.0149	35	7	200	1171
exp_15.2	570.000	0.0172	97	9	200	1939
exp_15.3	580.000	0.0169	85	9	200	1413
exp_15.4	690.000	0.0143	85	8	200	1294
exp_15.5	610.000	0.0161	89	9	200	1383
exp_15.6	660.000	0.0149	59	9	200	1480
exp_15.7	660.000	0.0149	73	9	200	1634
exp_15.8	640.000	0.0154	97	9	200	1787
exp_15.9	620.000	0.0159	75	7	200	1718
exp_16.0	520.000	0.0189	85	18	50	4512
exp_16.1	440.000	0.0222	89	16	50	4828
exp_16.2	440.000	0.0222	99	13	50	4929

exp_16.3	430.000	0.0227	99	17	50	5083
exp_16.4	470.000	0.0208	93	21	50	4667
exp_16.5	460.000	0.0213	97	13	50	4917
exp_16.6	440.000	0.0222	93	11	50	4837
exp_16.7	450.000	0.0217	93	13	50	4665
exp_16.8	400.000	0.0244	99	13	50	4708
exp_16.9	420.000	0.0233	99	10	50	4732
exp_17.0	640.000	0.0154	15	6	50	423
exp_17.1	680.000	0.0145	89	8	50	387
exp_17.2	630.000	0.0156	35	9	50	584
exp_17.3	560.000	0.0175	75	9	50	559
exp_17.4	660.000	0.0149	93	7	50	626
exp_17.5	610.000	0.0161	69	8	50	671
exp_17.6	640.000	0.0154	97	9	50	543
exp_17.7	630.000	0.0156	69	10	50	539
exp_17.8	650.000	0.0152	51	8	50	467
exp_17.9	680.000	0.0145	35	6	50	403
exp_18.0	460.000	0.0213	99	20	100	5525
exp_18.1	410.000	0.0238	99	15	100	5478
exp_18.2	470.000	0.0208	97	13	100	5674
exp_18.3	460.000	0.0213	93	14	100	5140
exp_18.4	460.000	0.0213	91	18	100	5639
exp_18.5	440.000	0.0222	97	12	100	5456
exp_18.6	380.000	0.0256	95	20	100	5613
exp_18.7	560.000	0.0175	55	14	100	5120
exp_18.8	370.000	0.0263	97	13	100	5068
exp_18.9	460.000	0.0213	95	13	100	4679
exp_19.0	640.000	0.0154	73	8	100	505
exp_19.1	650.000	0.0152	79	9	100	867
exp_19.2	680.000	0.0145	77	9	100	755
exp_19.3	630.000	0.0156	73	7	100	707
exp_19.4	690.000	0.0143	81	8	100	890
exp_19.5	620.000	0.0159	93	8	100	915
exp_19.6	630.000	0.0156	65	8	100	786
exp_19.7	660.000	0.0149	79	8	100	860
exp_19.8	610.000	0.0161	49	7	100	1089
exp_19.9	680.000	0.0145	95	9	100	799
exp_20.0	430.000	0.0227	95	14	150	5742
exp_20.1	450.000	0.0217	91	13	150	5619
exp_20.2	480.000	0.0204	79	14	150	5419
exp_20.3	470.000	0.0208	97	18	150	5532
exp_20.4	450.000	0.0217	89	16	150	5695

exp_20.5	460.000	0.0213	97	19	150	5625
exp_20.6	470.000	0.0208	91	15	150	5495
exp_20.7	540.000	0.0182	89	15	150	5574
exp_20.8	420.000	0.0233	93	17	150	5902
exp_20.9	480.000	0.0204	73	14	150	5693
exp_21.0	650.000	0.0152	51	9	150	1166
exp_21.1	630.000	0.0156	95	9	150	1032
exp_21.2	680.000	0.0145	95	9	150	1293
exp_21.3	640.000	0.0154	93	8	150	962
exp_21.4	650.000	0.0152	79	8	150	1456
exp_21.5	660.000	0.0149	29	8	150	727
exp_21.6	620.000	0.0159	63	7	150	1344
exp_21.7	530.000	0.0185	85	7	150	1409
exp_21.8	510.000	0.0192	91	9	150	1450
exp_21.9	580.000	0.0169	79	8	150	1135
exp_22.0	420.000	0.0233	87	10	200	6074
exp_22.1	510.000	0.0192	81	21	200	6161
exp_22.2	490.000	0.0200	95	12	200	6178
exp_22.3	560.000	0.0175	79	11	200	5996
exp_22.4	420.000	0.0233	99	16	200	5953
exp_22.5	450.000	0.0217	85	11	200	5022
exp_22.6	480.000	0.0204	93	20	200	5649
exp_22.7	500.000	0.0196	93	17	200	5660
exp_22.8	450.000	0.0217	93	13	200	5272
exp_22.9	360.000	0.0270	85	13	200	5494
exp_23.0	620.000	0.0159	93	9	200	2217
exp_23.1	610.000	0.0161	49	8	200	1277
exp_23.2	640.000	0.0154	43	6	200	1037
exp_23.3	620.000	0.0159	81	9	200	1150
exp_23.4	580.000	0.0169	91	8	200	1885
exp_23.5	590.000	0.0167	71	8	200	2062
exp_23.6	610.000	0.0161	65	9	200	1479
exp_23.7	670.000	0.0147	89	9	200	1326
exp_23.8	570.000	0.0172	39	7	200	1242
exp_23.9	650.000	0.0152	83	8	200	1514
exp_24.0	470.000	0.0208	97	12	50	7030
exp_24.1	470.000	0.0208	95	16	50	6491
exp_24.2	450.000	0.0217	95	13	50	5898
exp_24.3	420.000	0.0233	99	14	50	5851
exp_24.4	420.000	0.0233	97	15	50	6006
exp_24.5	490.000	0.0200	73	19	50	6003
exp_24.6	470.000	0.0208	97	17	50	5754

exp_24.7	390.000	0.0250	93	17	50	5871
exp_24.8	430.000	0.0227	91	14	50	5825
exp_24.9	480.000	0.0204	99	14	50	5945
exp_25.0	570.000	0.0172	85	8	50	604
exp_25.1	570.000	0.0172	69	8	50	489
exp_25.2	610.000	0.0161	85	9	50	561
exp_25.3	610.000	0.0161	79	7	50	594
exp_25.4	620.000	0.0159	87	8	50	593
exp_25.5	660.000	0.0149	57	7	50	508
exp_25.6	630.000	0.0156	89	8	50	546
exp_25.7	620.000	0.0159	93	8	50	540
exp_25.8	630.000	0.0156	93	9	50	627
exp_25.9	640.000	0.0154	79	8	50	576
exp_26.0	490.000	0.0200	89	14	100	6280
exp_26.1	460.000	0.0213	81	14	100	6321
exp_26.2	510.000	0.0192	89	22	100	6331
exp_26.3	460.000	0.0213	89	14	100	6195
exp_26.4	450.000	0.0217	89	13	100	6327
exp_26.5	450.000	0.0217	95	15	100	6300
exp_26.6	510.000	0.0192	91	17	100	6736
exp_26.7	400.000	0.0244	85	13	100	6303
exp_26.8	480.000	0.0204	91	14	100	6081
exp_26.9	500.000	0.0196	89	12	100	6367
exp_27.0	610.000	0.0161	87	9	100	586
exp_27.1	600.000	0.0164	73	7	100	736
exp_27.2	680.000	0.0145	47	8	100	691
exp_27.3	660.000	0.0149	93	8	100	1147
exp_27.4	640.000	0.0154	91	8	100	870
exp_27.5	650.000	0.0152	69	7	100	732
exp_27.6	660.000	0.0149	99	9	100	648
exp_27.7	550.000	0.0179	73	7	100	835
exp_27.8	640.000	0.0154	21	6	100	760
exp_27.9	650.000	0.0152	37	8	100	563
exp_28.0	360.000	0.0270	99	12	150	6775
exp_28.1	430.000	0.0227	97	14	150	6733
exp_28.2	460.000	0.0213	95	19	150	7178
exp_28.3	460.000	0.0213	95	18	150	6664
exp_28.4	460.000	0.0213	95	15	150	6829
exp_28.5	470.000	0.0208	99	20	150	6794
exp_28.6	480.000	0.0204	87	13	150	6508
exp_28.7	420.000	0.0233	93	11	150	6576
exp_28.8	450.000	0.0217	95	11	150	6923

exp_28.9	520.000	0.0189	69	12	150	6551
exp_29.0	690.000	0.0143	75	9	150	1482
exp_29.1	600.000	0.0164	97	9	150	1487
exp_29.2	680.000	0.0145	85	8	150	1297
exp_29.3	650.000	0.0152	75	8	150	679
exp_29.4	650.000	0.0152	77	9	150	1021
exp_29.5	550.000	0.0179	85	8	150	1183
exp_29.6	670.000	0.0147	43	7	150	816
exp_29.7	630.000	0.0156	89	10	150	1161
exp_29.8	650.000	0.0152	71	9	150	765
exp_29.9	630.000	0.0156	91	9	150	1246
exp_30.0	480.000	0.0204	93	14	200	7058
exp_30.1	410.000	0.0238	87	17	200	7004
exp_30.2	460.000	0.0213	93	19	200	6885
exp_30.3	400.000	0.0244	99	17	200	7292
exp_30.4	410.000	0.0238	99	14	200	6986
exp_30.5	410.000	0.0238	99	20	200	8831
exp_30.6	440.000	0.0222	95	12	200	8215
exp_30.7	450.000	0.0217	97	15	200	8371
exp_30.8	480.000	0.0204	99	21	200	10000
exp_30.9	410.000	0.0238	91	15	200	9611
exp_31.0	640.000	0.0154	33	5	200	1671
exp_31.1	660.000	0.0149	59	8	200	1256
exp_31.2	640.000	0.0154	55	7	200	1573
exp_31.3	660.000	0.0149	65	6	200	1729
exp_31.4	670.000	0.0147	95	9	200	1406
exp_31.5	620.000	0.0159	95	8	200	1561
exp_31.6	590.000	0.0167	81	8	200	1829
exp_31.7	650.000	0.0152	65	6	200	1045
exp_31.8	660.000	0.0149	77	8	200	1661
exp_31.9	700.000	0.0141	49	8	200	1253

- Resultados del experimento 3:

Repetición	Raw Fitness	Adj. Fitness	Best nodes rpb 0	Best depth rpb 0	Generations	Time (s)
exp_0.0	42	0,0233	101	8	100	2325845